VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Analýza komunikačních protokolů pro budoucí sítě

Analysis of Communications Protocols for Future Networks

Rok 2016 **Nikola Hutyrová**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Nikola Hutyrová**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Analýza komunikačních protokolů pro budoucí sítě
Analysis of Communications Protocols for Future Networks

Jazyk vypracování: čeština

Zásady pro vypracování:

S nárůstem počtu koncových zařízení a jejich začlenění do stávajících sítí jsou stále vyšší nároky na komunikační protokoly. Cílem bakalářské práce je zhodnocení síťových komunikačních protokolů, které budou využívány v budoucích sítích, jako je například internet věcí. Předpokládá se využití komunikačních protokolů pro přenos informací mezi koncovými zařízeními.

Řešení bude obsahovat:
1. Teoretický popis komunikačních protokolů pro internet věcí.
2. Instalaci a konfiguraci sítě pro datové přenosy, konfiguraci klientů a serveru.
3. Implementaci a testování vybraných protokolů, např. MQTT, XMPP, AMQP.
4. Výběr a zhodnocení vhodného komunikačního protokolu pro datové přenosy.

Seznam doporučené odborné literatury:

[1] Jeff Mesnil *Mobile and Web Messaging: Messaging Protocols for Web and Mobile Devices* O'Reilly Media 2014, ISBN-13: 978-1491944806
[2] Peter Waher *Learning Internet of Things* Packt Publishing 2015, ISBN-13: 978-1783553532
[3] Adrian McEwen, Hakim Cassimally *Designing the Internet of Things* Wiley 2013, ISBN-13: 978-1118430620

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Pavel Nevlud**

Datum zadání: 01.09.2015
Datum odevzdání: 29.04.2016

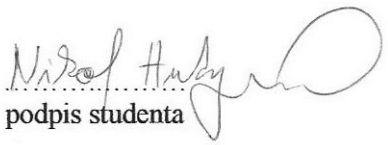
doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou/diplomovou práci vypracovala samostatně. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

V Ostravě dne: *28. dubna 2016*



podpis studenta

Poděkování

Ráda bych poděkovala vedoucímu bakalářské práce Ing. Pavlu Nevludovi za jeho ochotu a vstřícné jednání, odbornou pomoc, cenné rady a připomínky při konzultaci při vytváření této bakalářské práce.

Dále bych chtěla také poděkovat celé své rodině za neustálou podporu v průběhu celého studia.

Abstrakt

Tato bakalářská práce je věnována komunikačním protokolům budoucnosti, které se zaměřují a přizpůsobují v současné době modernímu a stále se rozšiřujícímu trendu Internetu věcí. Snaží se zasvětit do problematiky Internetu věcí a objasnit z jakého důvodu je nutné zkoumat komunikační protokoly k tomu určené. Popisuje srovnání aktuálních a nejčastěji používaných komunikačních protokolů, jejich technologii a princip činnosti, včetně možnosti jejich využití. Cílem této práce je zanalyzovat vybrané protokoly, porovnat jejich implementaci, podrobit je zátěžovým testům a z výsledných měření pomocí nasbíraných dat vyhodnotit, který z protokolů lze určit, jako perspektivní pro budoucí sítě.

Klíčová slova

Internet věcí, RFID, NFC, Thread, ZigBee, Z-Wave, Google Weave, Nest Weave, MQTT, STOMP, XMPP, AMQP, ZeroMQ, QoS, Instant messaging, broker, klient, zpoždění přenosu.

Abstract

This thesis is devoted to the communication protocols of the future that are focused and adapted to the present, modern and ever-increasing trend of Internet of Things. It tries to devote to the issue of Internet of things and explain why it is necessary to examine the communications protocols designated for this purpose. It describes a comparison of current and most commonly used communication protocols, their technology and principle of operation, including the possibility of their use. The aim of this work is to analyze the specific protocols, compare their implementation, subjecting them to stress tests and the resulting measurements using collected data to assess which of the protocols can be identified as promising for the future network.

Key words

Internet of Things, RFID, NFC, Thread, ZigBee, Z-Wave, Google Weave, Nest Weave, MQTT, STOMP, XMPP, AMQP, ZeroMQ, QoS, Instant messaging, broker, client, end-to-end delay.

Obsah

Seznam použitých zkratek

Zkratka	Význam
AES	Advanced Encryption Standard
AIM	AOL Instant Messenger
AMQP	Advanced Message Queuing Protocol
CPU	Central processing unit
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines Corporation
ICQ	I Seek you
IM	Instant messaging
IoT	Internet of Things
IRC	Internet Relay Chat
JID	Jabber ID
MAC	Media access control address
MQTT	Message Queuing Telemetry Transport
MSN	Microsoft network
NFC	Near Field Communication
OS	Operating system
PAN	Personal Area Network
QIP	Quiet Internet Pager
QoS	Quality of Service
RAM	Random Access Memory
RFID	Radio Frequency Identification
SASL	Simple Authentication and Security Layer
SCTP	Stream Control Transmission Protocol
SMS	Short message service
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture
STOMP	Simple/Streaming Text Oriented Message Protocol
TCP	Transmission Control Protocol

TLS	Transport Layer Security
UA	User Agent
UDP	User Datagram Protocol
UID	Unique Identification Number
USA	United States of America
UTF	UCS/Unicode Transformation Format
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol

Seznam ilustrací

Úvod

Nejen v běžném světě, ale i ve světě Informačních technologií, je komunikace neodmyslitelnou každodenní rutinou. K tomu, abychom byli schopni spolu denně komunikovat elektronickou formou, nestačí znát pouze správný jazyk, ale je nutné znát prostředek vyhovující našim potřebám, který stanoví přesná pravidla pro odesílání a přijímání elektronických zpráv. Tato potřeba otevírá dveře komunikačním protokolům, jejichž úkolem je vytvořit dohodu mezi komunikujícími stranami ve světě Informačních technologií.

V dnešní době se tato rutina rozšířila i do světa Internetu věcí, který otevírá nové možnosti a výzvy komunikačním protokolům. Zde se nejedná pouze už o komunikaci mezi lidmi, nebo pouze mezi stroji, zde se jedná o propojení komunikace mezi člověkem a zařízeními pomocí rozhraní, kde tyto takzvané "chytré věci" mají vlastní identitu a jsou schopny samy vyvozovat závěry, dle předaných znalostí, zkušeností a potřeb člověka. Zasazením do problematiky Internetu věcí se zabývá první kapitola této práce.

Druhá kapitola této práce se věnuje komunikačním protokolům určených pro Internet věcí. Popisuje dobře známé a často používané technologie, které bereme jako samozřejmé, ale také popisuje technologie, které už nejsou na první pohled patrné, uživatel často neví, co za nimi stojí a diskutuje se o nich, jako o technologiích budoucnosti.

Třetí kapitola zahajuje zpracování praktické části této práce, zabývá se konfigurací klientů a serveru vybraných komunikačních protokolů, často zmiňovaných, jako protokolů budoucnosti.

Čtvrtá, a sice poslední část této práce zachycuje testování těchto protokolů. Popisuje, jakým testům byly protokoly podrobeny, implementaci testů, kterých protokolů se testování týkalo, zobrazuje výsledky testů v grafech a srovnání výsledků protokolů. V závěru dojde ke zhodnocení a doporučení vybraných komunikačních protokolů.

1 Internet věcí (IoT)

V současné době často diskutované téma. S tímto pojmem se setkáváme doslova na každém kroku, nejen autoři se o něj opírají ve svých článcích a odborné literatuře, ale i různé společnosti v internetu věcí spatřily svůj potenciál a vycítily šanci prosadit se na trhu. Přes mnoho technických a provozních otázek, ale i skeptických otázek týkajících se bezpečnosti internetu věcí, se zdá, že panuje všeobecná shoda. Internet věcí slibuje vytvoření nové generace na síti Internet, a sice umožní uživatelům a sdíleným objektům sdílení hladkým, automatizovaným způsobem.

1.1 Zrození myšlenky IoT

Poté, co došlo k masivnímu a zásadnímu rozšíření sítě internet v devadesátých letech minulého století, především díky službě World Wide Web v rámci internetového prohlížeče, se lidstvo v prvním desetiletí tohoto století ocitlo v další fázi rozšíření internetu o mobilní sítě internet podporované mobilními zařízeními, včetně mobilních prohlížečů. Nyní stojíme na prahu další revoluce ve světě internetu, která spojuje objekty reálného světa se světem virtuálním a umožňuje jejich vzájemnou komunikaci kdekoliv a kdykoliv v rámci sítě internet. Internet věcí nám umožňuje propojení nejen s člověkem, ale s jakoukoli věcí v tomtéž prostoru a čase[1].

Pojem internet věcí poprvé zazněl z úst britského průkopníka v oblasti technologií Kevina Ashtona, který jej v roce 1999 jako asistent použil ve své prezentaci pro společnost Procter & Gamble, v souvislosti s novou myšlenkou zavedení RFID technologie v jejich dodavatelském řetězci. Později se tento muž stal spoluzakladatelem a výkonným ředitelem společnosti Auto-ID Center v Massachusetts Institute of Technology (MIT), kde vytvořili globální standardní systém pro technologii RFID, o níž budeme hovořit později, a širokou škálu identifikačních technologií[2].

1.2 Koncept a specifikace

Internet věcí radikálně mění způsob, jakým mohou společnosti přistupovat k řízení jejich majetku, sledování jejich obchodních operací a způsob jakým jednají se svými zákazníky.

Internet věcí můžeme koncepčně definovat jako dynamickou globální síťovou infrastrukturu s vlastními možnostmi konfigurace, založenou na standardních komunikačních protokolech a interoperabilitě, kde fyzické a virtuální "věci" mají vlastní identitu, fyzické atributy a virtuální personalizované používané inteligentní rozhraní a jsou integrovány do informační sítě[1].

Od internetu věcí se očekává, že se "chytré" věci, objekty a inteligentní výrobky stanou aktivními účastníky v oblasti obchodních, průmyslových, informačních a sociálních procesů, kde mohou vzájemně mezi sebou a též s prostředím interagovat, komunikovat a vyměňovat si data a informace citlivé na prostředí, zatímco reagují na skutečné události reálného světa a mají vliv na běžící procesy, které spouštějí události a vytváření služeb s nebo bez přímé účasti lidského činitele[1].

Služby budou moci interagovat s těmito "chytrými" věcmi, objekty, výrobky pomocí standardních rozhraní, která budou poskytovat potřebná spojení prostřednictvím sítě internet, na dotazování a změnu svého stavu a získání veškerých souvisejících informací s přihlédnutím k bezpečnosti a ochraně soukromí[1].

1.3 Vývoj a prognóza

Koncept IoT byl přijat Evropskou Unií a publikován v březnu roku 2007. V dubnu roku 2008 byla publikována Národní radou USA zpráva o šesti technologiích s předpokládaným vlivem okolo roku 2025, která zahrnovala i technologii internetu věcí[1].

Ve druhé polovině roku 2009 byla přednesena řada významných veřejných projevů na téma týkající se problematiky IoT v Číně. Dne 7. srpna téhož roku čínský premiér Wen Ťia-pao vyzývá k rychlému rozvoji internetových technologií v oblasti IoT a uvádí zajímavou rovnici[1]:

Internet + Internet věcí = Moudrost Země

Harald Sundmeaker a spol. ve své publikaci z roku 2010 zveřejnil celosvětový výzkum, který uvádí, že v roce 2010 se na světě vyskytovalo přibližně 1,5 miliardy počítačů s internetovým připojením a přibližně 1 miliarda mobilních telefonů připojených k internetu[3]. Tento stav před pěti lety nazvali autoři publikace jako "internet počítačů" a dovolili si odhadnout množství připojených zařízení současné doby "internetu věcí", a sice 50 až 100 miliard zařízení s internetovým připojením do roku 2020[3]. Nyní máme možnost porovnat, do jaké míry se odhad lišil se současným přibližným stavem a současnými odhady množství připojených různých zařízení do budoucna.

Společnost Gartner v roce 2015 zveřejnila svůj průzkum, se kterým se shodují i společnost AVG a Cisco. Výsledky ukazují, že v roce 2015 bylo k internetu připojeno přibližně 5 miliard zařízení, avšak podle výše uvedených společností se internet věcí nebude do roku 2020 těšit takovému rozmachu, jak odhadovali Sundmeaker a spol., a internet věcí v roce 2020 by měl s nadsázkou pojmout pouze přibližně 25 až 50 miliard různých zařízení s internetovým připojením[4].

1.4 Možnosti využití IoT

Internet věcí je více než komunikace, IoT též vybavuje jednotlivé objekty inteligencí. Tato inteligence může být umístěna přímo na objektu samostatně v podobě příslušného mikročipu, nebo tím, že je objekt reprezentován infrastrukturou umístěnou na serveru internetu věcí, ke kterému může být objekt trvale nebo dočasně připojen[1].

V kontextu internetu věcí mají důležitou úlohu čidla, snímače a senzory. V rámci IoT můžeme předpokládat jejich použití např. při měření teploty, rychlosti, zrychlení, umístění souřadnic, spotřeby energie, půdních chemických parametrů, znečištění ovzduší aj. Mohou zajistit měření kritických parametrů např. letadel, mostů, budov a parametrů našeho životního prostředí[1]

Očekává se, že zvláště odvětví průmyslu, průmyslové výroby, automobilového průmyslu, zdravotnictví a letectví může v širokém měřítku využít právě internet věcí.

Co se týče letectví, je nejdůležitějším faktorem bezpečnost. Možnost bezdrátového monitorování jednotlivých sekcí letadla, jak vnitřních, tak i venkovních, by poskytla údaje, které mohou být dále napojeny na stávající systémy. Vznikem takovéto sítě by bylo možné sbírat data např. tlak, vlhkost, teplota aj., potřebná k analýze a vyvozování správných závěrů. Sběr dat by mohl přispět k plánování údržby a přehodnocení, či optimalizaci spotřeby energie při jednotlivých procesech[3].

Ve výrobním průmyslu internet věcí nabízí nové možnosti, jak být kreativní. Např. inteligentní nápojové automaty, které jsou připojeny k dodavatelskému řetězci a jsou schopny vnímat rovnováhu mezi nabídkou, zásobami a spotřebou[5].

Svět nezůstane pouze u inteligentních koncových světel automobilů a technologie "chytrého" parkování aj., ale díky internetu věcí, bude umožněno vytvářet celé inteligentní autonomní vozidla, která mohou změnit dynamiku dopravy a logistiky města. Inteligentní systémy osvětlení mohou změnit spotřebu napájení města. Může se však jednat i o digitální výdobytky našeho každodenního života, směřující k "chytré" domácnosti. Ať jde např. o osvětlení či termostat ovládané z "chytrého" telefonu, nebo lednici, která umí sestavit nákupní seznam a dokonce i sama zaslat objednávku přes internet[5].

Potenciál internetu věcí do budoucna tkví i oblastí bankovnictví, což nám může umožnit přístup k bankovnictví, bez ohledu na zařízení. "Chytré" hodinky či brýle, "chytrá" televize nebo lednice mohou sloužit přímo jako platební terminál. Avšak v hlavní roli je analýza klientských dat v reálném čase jako je, okamžitá nabídka při návštěvě obchodníka např. vcházíme k prodejci a na telefonu se nám objeví hlášení banky, jak vysokou půjčku si můžeme dovolit. V případě půjčky, nás může samotný vůz upozorňovat na blížící se splátky. Nemusí se ale vždy jednat o stěžejní situace, i taková automatická identifikace při vstupu na pobočku, pomůže bance vyhodnotit, jde-li o stávajícího klienta či nového, patřičně spravovat fronty a prioritizovat obsluhu tak, aby byla spokojenost na obou stranách a klient tak měl informace o aktuální čekací době na pobočkách[6].

1.5 Perspektiva a budoucnost

Perspektiva a budoucnost sítě internet, založená na standardních komunikačních protokolech, předpokládá sloučení, integraci zvlášť označovaných skupin s infrastrukturou jako internet věcí, internet osob (IoP), internet energetiky (IoE), internet médií (IoM) a internet služeb (IoS) pro globální, společnou informačně-technologickou platformu aplikačně propojených sítí spojující propojené "chytré" věci, objekty, produkty, výrobky, zařízení a mnoho dalších[1].

Internet energetiky je definován jako dynamická síťová infrastruktura, které propojuje energetickou síť se sítí internet a umožňuje odesílat informace o energetických jednotkách, blocích a jimi generovaných energetických tocích[1].

Internet služeb poskytuje služby v souvislosti s architekturou orientovanou na služby (SOA), webovými službami, podnikovou interoperabilitou, sémantickým webem a skládáním služeb, při současném zlepšování spolupráce mezi poskytovateli služeb a uživateli[1].

Internet médií se bude zabývat problematikou škálovatelného video kódování a 3D prostorovým video zpracováním, dynamickým přizpůsobováním se podmínkám v síti, které vyvolávají inovativní aplikace, jako jsou více hráčské mobilní hry, mobilní digitální kino a virtuální svět aj[1].

Internet osob propojuje rostoucí populaci uživatelů při současné podpoře jejich on-line aktivity a zachování volné výměny myšlenek, též poskytuje prostředky k usnadnění každodenního života lidí, obcí, organizací, což zároveň usnadňuje podnikání a odstraňuje bariéry mezi informacemi od výrobce a informacemi od spotřebitele, uživatele[1].

Rozšíření IoT spolu s dalšími rozvíjejícími se skupinami, jako výše zmíněnými, jsou základem digitálního hospodářství, digitální společnosti a tvoří základy pro budoucí ekonomiku založenou na znalostech a inovacích společnosti. Zařízení jako jsou "chytré" telefony, výrobky, produkty, věci, zařízení, stroje a komunikace mezi nimi budou hlavní hnací silou pro další rozvoj internetu věcí[1].

2 Komunikační protokoly pro internet věcí

2.1 RFID

V souvislosti s internetem věcí rozhodně stojí za zmínku protokol RFID neboli Radio Frekvenční Identifikace, jehož kořeny sahají do doby druhé světové války, kdy v 30. letech minulého století byl vyvinut první aktivní systém rádiové identifikace, předchůdce RFID, zabudovaný v radarech protivzdušné obrany.

2.1.1 Specifikace

Jedná se o jednu z nejvíce se rozvíjejících technologií, která slouží k označování objektů, sběru dat a poskytování objektivních informací o objektech v reálném čase[9].

Tuto technologii lze najít v různých odvětvích průmyslu jako je kontrola výrobních procesů, logistika, dodávky a expedice, ochrana evidence majetku, evidence zavazadel a osob, v obchodních řetězcích, v automobilovém průmyslu, v odvětví bezpečnosti, ale i v identifikaci zvířat[7].

Mezi hlavní výhody RFID patří bezkontaktní povaha, která nevyžaduje pro identifikaci objektu jeho přímou viditelnost, ani přesné polohování, dále rychlost čtení, přinášení nových možností funkcionality identifikačního procesu díky novým aktivním technologiím a ani špatné optické či atmosférické podmínky nebrání přenosu dat z čipu[8].

2.1.2 Technologie

RFID systém se skládá ze základních prvků, kterými jsou transpondéry, čtecí zařízení a řídící software.

Transpondér neboli RFID tag, je tvořen čipem, anténou a případně vlastním zdrojem energie, baterií, pokud se jedná o aktivní RFID tag. Pasivní tag vlastní zdroj energie nemá a je závislý na dodávce energie z antény čtecího zařízení, která šíří elektromagnetické pole. Další variantou může být i semipasivní tag, který má interní napájecí zdroj soužící k napájení integrovaných obvodů. Hlavní funkcí tagu, je uložení dat do vnitřní paměti a poskytnutí uložených údajů RFID systému. Komponenty mohou být zapouzdřeny a tag může mít podobu např. kreditní karty, "chytré" etikety, skleněného tagu[8].

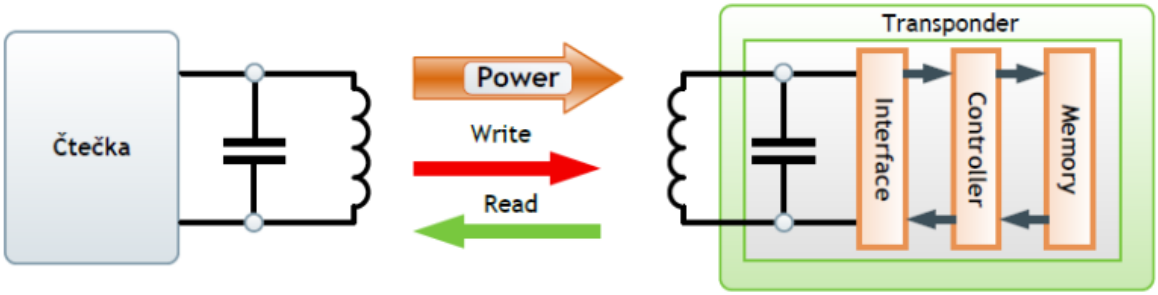
Čtecí zařízení je tvořeno vysílacím/přijímacím obvodem s dekodérem, anténou. V některých případech může být vybaveno i vlastním operačním systémem se základní funkcionalitou[8].

Řídící software neboli middleware, představuje software pro správu, filtraci a analýzu dat získaných z populace tagů. Mezi jeho funkce patří: zprostředkovat komunikaci s jednotlivými čtecími zařízeními a zpracovat získaná data., filtrovat získaná data, výsledek uchovávat v databázi a poskytovat je přes stanovené rozhraní dalším aplikacím[8].

Systémy RFID využívají rádiových vln, které pracují na různých vlnových délkách, a tedy mohou nebo nemusí projít různými druhy materiálů. Volba vhodné frekvence pro konkrétní aplikaci je jedna z nejdůležitějších fází návrhu řešení systému RFID. Platí, že čím vyšší frekvence, tím rychlejší přenos dat, ale zároveň delší vzdálenost, ve které je RFID čtecí zařízení schopno komunikovat s RFID tagem avšak za cenu větší citlivosti na přítomnost problematických materiálů[8].

2.1.3 **Princip činnosti**

Pro zjednodušení bude popsán obecný princip činnosti pasivního RFID tagu. Čtecí zařízení prostřednictvím antény vysílá periodicky na svém nosném kmitočtu elektromagnetickou vlnu do okolí. Objeví-li se ve vhodné vzdálenosti od antény tag, který je naladěn na stejnou frekvenci, je tato vlna přijata anténou tagu. Indukované napětí na anténě tagu vyvolá střídavý elektrický proud, který je usměrněn a nabíjí kondenzátor v tagu. Uložená energie je použita pro napájení logických a rádiových obvodů tagu. Když napětí na kondenzátoru dosáhne minimální potřebné úrovně, spustí řídící obvody uvnitř tagu a ten začne odesílat odpověď čtecímu zařízení. Vysílání tagu je realizováno zpravidla pomocí dvoustavové ASK modulace, která je realizována změnou zakončovací impedance antény transpondéru. Modulace představuje důkladné ovlivňování tří parametrů signálu, a to je výška, frekvence a fáze amplitudy. Pomocí modulace vlny vysílané ze čtecího zařízení lze do tagu i zapisovat. Analýzou těchto vln kdekoliv v dosahu čtecího zařízení můžeme zpětně zrekonstruovat zprávu přijaté vlny - demodulace. Odrazy, které vznikají změnou impedance antény, jsou detekovány čtecím zařízením a interpretovány jako logické úrovně 1 a 0[8].



Obrázek 2.1: Základní schéma komunikace v RFID[10]

Dostatečná energie pro nabití kondenzátoru v transpondéru a schopnost detekovat přijatou odpověď transpondéru čtecím zařízením jsou tak hlavní hardwarové podmínky fungování RFID systému. S rostoucí vzdáleností mezi čtecím zařízením a transpondérem postupně klesá kvalita RFID signálu. Nárůst šumu v základním signálu vede až k nemožnosti úspěšné detekce přijaté zprávy[8].

2.2 NFC

Technologie NFC neboli Near Field Communication v jistém smyslu vychází z předchozí technologie RFID, avšak nikoli technologickým způsobem fungování, ale posouvá o kus dále samotný princip bezkontaktní komunikace.

Jak už název napovídá, jedná se o technologii, která umožňuje dvěma elektronickým zařízením bezdrátově komunikovat na velmi krátkou vzdálenost v řádech centimetrů za předpokladu, že se komunikace účastní alespoň jedno aktivní zařízení. Pokud dojde ke spojení dvou aktivních zařízení, mohou si informace vyměňovat vzájemně. V případě spojení aktivního a pasivního zařízení probíhá komunikace pouze od pasivního zařízení k aktivnímu[11].

Technologií NFC lze přenášet téměř všechny druhy dat včetně obrázků a mp3 souborů, z toho důvodu NFC stanovuje tři různé režimy vysílání: peer-2-peer (režim klient-klient), režim pro čtení-zápis a režim emulace karty[11].

Na rozdíl od RFID disponuje NFC inovacemi jako podpora šifrování, kterou zprostředkovává Secure element, což je nezávislý čip s úložištěm dat, sloužící jako most mezi NFC čipem a infrastrukturou telefonu[12]. Další inovací NFC je podpora vzájemné oboustranné komunikace dvou aktivních zařízení. Díky tomu našlo NFC uplatnění především u bezkontaktních plateb nejen platebními kreditními kartami, ale i pomocí chytrých telefonů, které disponují touto technologií a mohou tak být využívány jako jízdenky MHD, vstupenky nebo věrnostní karty[13].

2.3 Thread

Tento otevřený bezdrátový síťový protokol byl uveden v polovině roku 2014 po vzájemné spolupráci společností Google's Nest Labs, Samsung Electronics, ARM za účelem podpory a rozšíření internetu věcí do "chytrých" domácností pro nadcházející roky[14].

Jedná se tedy o protokol navržený speciálně pro domácnosti tak, aby podporoval širokou škálu výrobků a spotřebičů od ovládání klimatizace až po hospodaření s energií a zabezpečením domácnosti. Umožňuje připojit až 250 zařízení v rámci jedné domácnosti. Thread je založen na standardu 802.15.4 pro rádiovou komunikaci a jeho využití pro příští generace potvrzuje skutečnost, že nativně zpracovává protokol IPv6 pro 6LoWPAN[15].

Vyznačuje se energeticky úsporným provozem, umožňuje i zařízením na bateriový provoz být součástí sítě a také hardwarovou kompatibilitou s dalšími technologiemi. Díky podpoře tzv. ospalých uzlů, umožňuje několik let provozu zařízení, které funguje i na jedné AA baterii[16].

Instalace sítě je snadná a intuitivní, pro uživatele jednoduchá k použití, jelikož poskytuje příjemné uživatelské prostřední, kde pomocí chytrých telefonů a počítačů mohou přidávat či odebírat zařízení[15].

Všechny Thread sítě jsou bezpečně šifrovány, bezpečnostní díry nalezeny u jiných bezdrátových protokolů jsou uzavřeny pomocí AES šifry a k síti se lze připojit pouze autorizovanými přístroji. Thread síť je založena na principu mesh sítě, což znamená, že nabízí robustní samoléčení sítě bez jediného bodu (zařízení) selhání[15].

2.4 ZigBee

Zigbee je bezdrátová technologie určená pro komunikaci nízko výkonových zařízení a přenos informací na krátké vzdálenosti přibližně do 75 metrů v sítích PAN. Jedná se o otevřený standard založený na specifikaci IEEE 802.15.4, jehož účelem bylo konkurovat technologii Bluetooth, platný od listopadu roku 2004[17].

Tato technologie byla navržena tak, aby vyplnila mezeru mezi technologií Bluetooth a Wi-Fi. Hlavní výhodou ZigBee je velmi nízká spotřeba energie, neboť podporuje i zařízení s bateriovým provozem a nízká pořizovací cena včetně jednoduchosti implementace. Protokoly ZigBee jsou z hlediska jednoduchosti navrženy tak, aby mohly být zpracovatelné i 8bitovými kontroléry pro nejjednodušší aplikace[17]. Za tyto výhody ZigBee zaplatilo svou daň v podobě nízké přenosové rychlosti přibližně od 40kbit/s do 250kbit/s, jinými slovy se pomalejší ZigBee uplatní u bezdrátových sítí, u nichž není požadován přenos velkého objemu dat[18].

Díky různorodosti předpokládaných aplikací standard definuje tři základní režimy přenosu dat: periodicky se opakující, nepravidelné přenosy a opakující se přenosy, u nichž je požadavek na malé zpoždění. Jedná se o nízkofrekvenční protokol, tudíž nepracuje na bázi IP adres jako např. Thread[17].

ZigBee také definuje tři různé síťové topologie, přičemž základní topologií je hvězdicová topologie s centrálním řídícím uzlem. Dalším typem je stromová struktura nebo síť mesh[17].

Tato technologie našla uplatnění v odvětví průmyslové automatizace a automatizace budov např. kontrola přístupu a je využíváno i v počítačových perifériích[18].

2.5 Z-Wave

Bezdrátový komunikační protokol navržen pro účely domácí automatizace, snažící se minimalizovat spotřebu energie. Technologií je velmi podobný ZigBee a mají také mnoho společného.

Z-Wave, stejně jako ZigBee, je nízkofrekvenční protokol a nepracuje na bázi IP adres, ale ani neinterferuje s Wi-Fi sítí. Každý z těchto protokolů definuje všechny síťové funkce všech kompatibilních zařízení pro zajištění maximální kompatibility s jinými zařízeními s tímto protokolem, avšak vzájemně ZigBee a Z-Wave kompatibilní nejsou. Oba protokoly používají k zabezpečení komunikace symetrické šifrování AES-128, stejně jako nabízejí některá internetová bankovnictví[14].

Z-Wave pro svou architekturu používá mesh topologii sítě, kde každá domácnost má vlastní ID sítě a kde je každému prvku (zařízení) přiřazeno identifikační 32bitové číslo[14].

Jako jedna z mála technologií si udržuje zpětnou kompatibilitu napříč všemi verzemi s jakýmkoli jiným produktem Z-Wave, jinými slovy, když je nějaký produkt certifikován jako Z-Wave, bude schopen fungovat s jakýmkoli jiným Z-Wave produktem z minulosti, přítomnosti i budoucnosti[14].

Využití této technologie v domácnosti je opravdu všestranné. Z-Wave technologii lze integrovat i do stávajících zabezpečovacích systémů domácnosti, což je pro zákazníka po ekonomické stránce velmi výhodné a ušetří tak použitím stejných senzorů pro zabezpečení a automatizaci domácnosti. Zařízení Z-Wave obsahují i předkonfigurované Z-Wave moduly např. k ovládání osvětlení, nebo pokročilý programovací on-line plánovač, které lze dálkově aktivovat či deaktivovat. Z-Wave zařízení dokonce dokážou automatizovat zavlažovací systém na základě internetové předpovědi počasí[19].

	Z-Wave	ZigBee	WeMo	Thread
Operating range	100 feet	35 feet	100 feet	100 feet (theoretical)
Max no. devices	232	65,000	Router-dependent	250-300
Data rate	9.6-100 kbps	40-250 kbps	Router-dependent	250 kbps
Frequency	908/916 MHz (U.S.)	915 MHz/2.4 GHz	2.4 GHz	2.4 GHz
Network type	Mesh	Mesh	Star	Mesh
Needs hub?	Yes	Yes	No	Yes

Obrázek 2.2: Srovnání vlastností technologicky blízkých protokolů[20]

2.6 (Google) Weave a Nest Weave

Weave je program společnosti Google, s cílem podpořit připojená zařízení všeobecně v rámci prostoru Internetu věcí. Je to komunikační vrstva, díky které mohou inteligentní domácí zařízení spolu komunikovat, ve spojení s operačním systémem Brillo, navrženým pro drobnou nepočítačovou elektroniku IoT. Nest Weave je vlastní aplikační protokol společnosti Nest Labs, současně používán Nest produkty po celém světě[21].

2.6.1 Weave technologie

Weave síť se skládá z několika bodů (zařízení), kde každý bod má své jméno, přezdívku a unikátní identifikátor UID, který je po každém spuštění zařízení změněn. Weave směrovače sestavují s každým účastníkem sítě šifrované TCP spojení, potřebné k výměně informací o topologii sítě. Naopak účastníci sítě navazují spojení pomocí UDP protokolu, které může být taktéž šifrováno a slouží pro přenos paketů. Weave vytváří síťový most na hostitele, kde každý účastník je napojen na tento most přes svou IP adresu a masku podsítě. Weave směrovač zachycuje jednotlivé pakety přes vytvořený most pomocí UDP protokolu od účastníků sítě a je schopen díky MAC adrese účastníka a znalosti topologie sítě směrovat a předat pakety konkrétnímu účastníkovi. Informace o změnách v topologii sítě si účastníci mezi sebou vyměňují pomocí TCP protokolu[22].

2.6.2 Specifikace Nest Weave

Protokol vytvořen pro chytré domácnosti, umožňující produktovou integraci pro inteligentní domácí zařízení jako jsou např. Nest inteligentní termostat, Nest detektor kouře, Mimo Baby monitor aj. Klíčový rozdíl této nové integrace spočívá v schopnosti zařízení komunikovat použitím Nest Weave protokolu přímo s Nest zařízeními bez nutnosti Wi-Fi připojení napříč sítí v domácnosti, což zajišťuje větší spolehlivost, stabilitu a bezpečnost domácí sítě i v případě selhání Wi-Fi sítě[21].

2.7 MQTT

MQTT, celým názvem Message Queue Telemetry Transport, je otevřený komunikační protokol, původně navrhován a vyvinut společností IBM a Eurotech pro nespolehlivé sítě s nízkou šířkou pásma a vysokou latencí, později věnován organizaci OASIS[23].

2.7.1 Specifikace

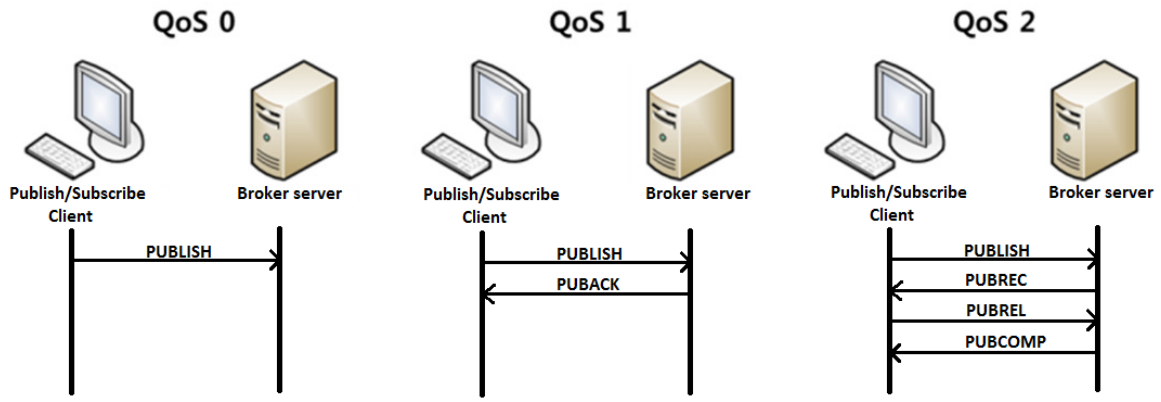
MQTT byl navržen tak, aby byl otevřený, jednoduchý, spolehlivý a snadno implementovatelný. Skládá se z jednoho Broker serveru (zprostředkovatele) a dvou typů klientů nazývaných jako Publisher (veřejný klient/vydavatel) a Subscriber (odběratel). Broker server funguje jako prostředník pro posílané zprávy mezi těmito dvěma klienty[23]. Existuje několik MQTT Broker serverů dostupných jako RabbitMQ, Mosquitto, ActiveMQ, Apollo aj. [26]

Záhlaví MQTT zprávy obsahuje mimo jiné dva hlavní artefakty definované jako typ zprávy a úroveň QoS. Typ zprávy definuje metody, které indikují požadované akce. V podstatě je můžeme rozlišovat jako žádosti např. CONNECT, DISCONNECT, PUBLISH, SUBSCRIBE aj. a potvrzení např. CONNACK, PUBACK, SUBACK aj.[24]

2.7.2 Quality of Services (QoS)

MQTT podporuje tři úrovně QoS[23]:

- Úroveň 0 - zpráva Publish/Subscribe klienta je poslána pouze jednou po zprávě distribuce toku a není kontrolováno, zda zpráva úspěšně dorazila na místo určení, a tedy může cestou dojít k jakékoliv ztrátě, včetně ztráty zprávy.
- Úroveň 1 - zpráva Publish/Subscribe klienta je poslána alespoň jednou a je kontrolován stav doručení zprávy pomocí zprávy PUBACK, která potvrzuje doručení od klienta. Pokud dojde ke ztrátě PUBACK zprávy, může Broker server posílat tuto stejnou zprávu do doby, než obdrží potvrzení o doručení.
- Úroveň 2 - zprávy jsou předávány prostřednictvím použití právě jednoho 4-way handshake, tudíž není možné, aby došlo ke ztrátě zprávy na této úrovni, ale vzhledem k složitosti procesu je možné mít relativně dlouhé zpoždění mezi koncovými zařízeními (end-to-end).



Obrázek 2.3: Přenos paketu z pohledu úrovní QoS[23]

2.7.3 Proces přenosu zpráv

Role Publish klienta spočívá v připojení se k Broker serveru a publikování obsahu. Naopak role Subscribe klienta spočívá v připojení se k stejnému Broker serveru a odebírání obsahu, o který má zájem.

Publisher notifikuje Broker server s konkrétním tématem, které zamýšlí publikovat. Broker server si udržuje toto publikované téma, dokud si Subscriber toto téma nevyžádá. Vzhledem k tomu, že Broker server funguje jako zprostředkovatel mezi klienty, bude Subscriber klientovi doručena zpráva, kterou si na toto téma vyžádal[23].

Protokol je dnes implementován v mnoha odvětvích. Můžeme jej nalézt v nemocnicích, které jej používají pro komunikaci s kardiostimulátory apod., nebo u běžných uživatelů majících zálibu v sociálních sítích, neboť mobilní aplikace Facebook Messenger využívá aspektů MQTT. I producenti plynu a nafty našly využití tohoto protokolu, s jehož pomocí monitorují tisíce kilometrů potrubí[25].

2.8 STOMP

Ve volném překladu známý jako jednoduchý textově-orientovaný protokol, který vznikl z potřeby propojení podnikových zpráv ze skriptovacích jazyků např. Python, kde se provádějí typicky logické jednoduché operace. Jedná se o jednoduchý interoperabilní protokol navržený pro asynchronní zasílání zpráv mezi klienty prostřednictvím zprostředkujících serverů[28].

Jeho jedinečnost spočívá v tom, že poskytuje interoperabilní formát, který umožňuje STOMP klientům komunikovat s jakýmkoli message brokerem podporujícím protokol, je tedy jazykově nezávislý a multiplatformní[27].

I když je STOMP především textově-orientovaný protokol, umožňuje i přenos binárních zpráv[27]. Je založen na rámcích po vzoru HTTP protokolu a jeho výchozí kódování je UTF-8. Mimo to byl STOMP navržen tak, aby byl snadno implementovatelný jak na straně klienta, tak i na straně severu v širokém rozsahu jazyků[28].

STOMP server je modelován jako soubor cílů, ke kterému mohou být zprávy odeslané. Protokol zachází s destinací jako s neprůhledným řetězcem a její syntaxe je specifická dle implementace serveru. Tento protokol nedefinuje, jaká by měla být doručovací sémantika, doručovací sémantika se liší od serveru na server, a dokonce i z místa určení do místa určení. Serveru s podporou STOMP protokolu je tak umožněno být se sémantikou kreativní[28].

Klient a server spolu vzájemně komunikují pomocí STOMP rámců posílaných přes stream. STOMP klient je User-Agent, který může působit, případně i současně, ve dvou režimech[28]:

- Producent, který odesílá zprávy na místo určení na serveru přes SEND rámec.
- Spotřebitel, který odesílá SUBSCRIBE rámec pro danou destinaci a příjmá zprávy ze serveru jako MESSAGE rámce.

SEND rámec odesílá zprávu do místa určení v systému zasílání zpráv. V záhlaví rámce musí byt povinně vyplněna položka destination, která udává cílovou destinaci. Tělo SEND rámce je samotná zpráva, kterou máme v úmyslu odeslat[28].

SUBSCRIBE rámec slouží k registraci a poslechu dané destinace a stejně jako předešlý rámec vyžaduje destination. Veškeré zprávy přijaté na SUBSCRIBE cíle, budou dále doručovány jako MESSAGE rámce ze serveru na klienta[28].

2.9 XMPP

Ve volném překladu se jedná o rozšířitelný protokol pro posílání zpráv a zjištění stavu. XMPP je implementací obecného značkovacího jazyka XML. Vznikl na základě původního zastaralého protokolu Jabber. Nyní je Jabber komunikační platforma/server založena na protokolu XMPP a jeho rozšířeních. Hlavním důvodem vzniku XMPP, bylo vytvořit jazyk/protokol, který umožní sjednotit komunikátory a doručovat zprávy v skoro-reálném čase[31].

2.9.1 Specifikace

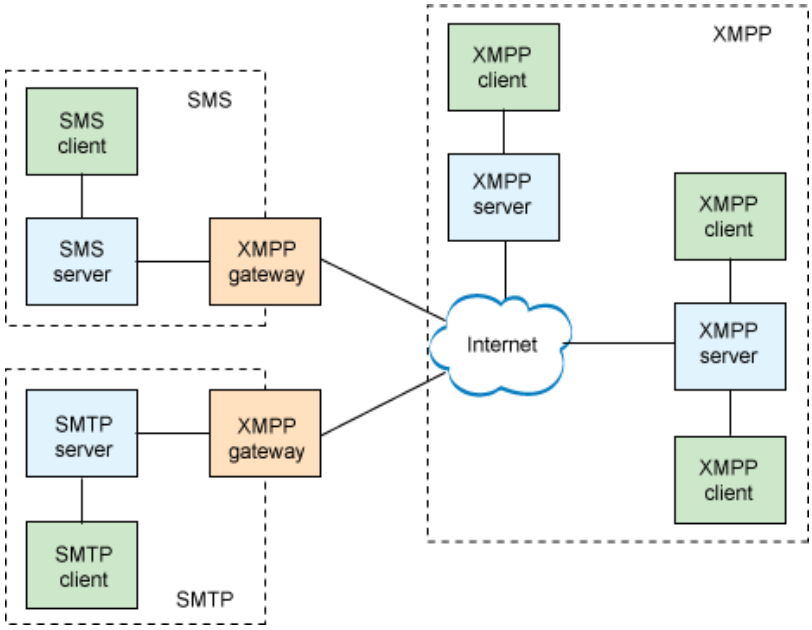
XMPP protokol se stal nejpopulárnějším protokolem pro tzv. Instant messaging(IM). Jeho hlavní výhodou oproti jiným protokolům je otevřenost a nezávislost na požadavcích jedné konkrétní firmy, je nezávislý na použité platformě. Díky tomu existuje celá řada klientů např. Miranda, iChat, QIP Infium aj. pro nejrůznější operační systémy a platformy i pro mobilní zařízení[31].

XMPP podporuje také tzv. transporty, což jsou služby Jabber serveru, umožňující uživatelům využívat i jiné služby, které přímo nejsou součásti XMPP protokolu, to znamená, že je možné z jednoho klientského programu komunikovat s uživateli jiných sítí např. ICQ, AIM, MSN aj.[31]

2.9.2 Architektura

Síť XMPP je založena na architektuře klient-server a není centralizována do jednoho místa jak je zvykem u většiny IM protokolů, ale je distribuovaná na servery po celém světě, na kterých je možné založit uživatelský účet, a tedy uživatel má svobodu volby serveru. Servery XMPP protokolu běží standardně na TCP portu 5222. Pro vzájemnou komunikaci serverů je pak vyhrazen port 5269[32].

Servery spolu také komunikují za účelem směrování mezi doménami. Vzdálené XMPP brány často existují za účelem překladu zahraničních zpráv mezi cizími doménami a protokoly, jako např. SMS, SMTP, IRC aj. XMPP je tak ideálním páteřním protokolem k poskytování konektivity mezi různými koncovými protokoly[30].



Obrázek 2.4: XMPP architektura včetně XMPP brány[29]

2.9.3 Jabber ID

Jedná se o identifikátor uživatele a je zkracován jako JID. Syntakticky i sémanticky je podobný e-mailovým adresám, tedy splňuje formát uživatel@server. XMPP umožňuje uživateli být přihlášen na svůj účet z více míst najednou. Kontakty jsou uloženy na serveru a uživatel se připojuje vždy pouze ke svému serveru, neboť jenom tento server je schopen ověřit jeho identitu[32].

2.9.4 Proces přenosu zpráv

Aby mohlo dojít ke komunikaci dvou uživatelů, je nutné, aby znali vzájemně své JID. Průběh spojení je zahájen prvním uživatelem, který napíše zprávu a odešle ji druhému uživateli, což znamená, že XMPP klient prvního uživatele pošle svoji zprávu na svůj server, kde je první uživatel zaregistrován. Server prvního uživatele otevře spojení se serverem druhého uživatele, kde je druhý uživatel registrován a předá mu zprávu. Server druhého uživatele doručí zprávu XMPP klientovi druhého uživatele. Pokud druhý uživatel není on-line bude zpráva uschována na jeho serveru a doručena později[32].

2.10 AMQP

Jedná se o otevřený standard aplikační vrstvy pro předávání obchodních (business) zpráv mezi aplikacemi a organizacemi, který spojuje systémy a zásobuje podnikové procesy informacemi, které potřebují a spolehlivě přenáší instrukce, které dosahují svých cílů[33].

AMPQ protokol vznikl v roce 2003 ve Velké Británii a byl vyvíjen skupinou 23 společností, která byla později reorganizována do členské sekce OASIS. Nejvýznamnější a nejpoužívanější aktuální verze AMPQ 1.0 byla spuštěna ke konci roku 2011[35].

2.10.1 Specifikace komunikace

AMQP je binární protokol a je navržen tak, aby účinně podporoval širokou škálu aplikací určených ke komunikaci a komunikačních vzorů. Cílem AMQP je umožnit aplikacím posílat zprávy prostřednictvím brokerů a přenést na ně tak zodpovědnost o doručení, navíc může být aplikován i v systémech peer-to-peer.

Komunikace může být iniciována jakýmkoliv AMQP klientem a probíhá přes TCP protokol, ale je možné vyjednat i použití alternativních transportních protokolů např. SCTP, UDP aj. Autentifikace a šifrování probíhá na bázi SASL a TLS[34].

Poskytuje řízený průtok komunikace s třemi druhy garance doručení zprávy[35]:

- Nejvíce jednou (at-most-once), tj. každá zpráva je doručena pouze jednou nebo nikdy.
- Nejméně jednou (at-least-once), tj. je jisté, že každá zpráva bude doručena alespoň jednou nebo může být dodána vícekrát.
- Právě jednou (exactly-once), tj. zpráva bude doručována pouze jednou s jistotou, že dorazí.

AMQP poskytuje mechanismy pro obnovení přenosu zpráv, pokud dojde ke ztrátě spojení např. při selhání služby. Dále poskytuje velmi schopný mechanismus řízení toku zpráv, který umožňuje spotřebitelům zpráv zpomalit producenty na přijatelnou a zvládnutelnou rychlost a umožňuje paralelně zpracovávat různé úlohy při různých rychlostech přes jedno spojení. AMQP umožňuje více konverzací přes jedno TCP spojení a definuje způsob zpracování skupin zpráv v rámci transakce. Technologie AMQP je multiplatformní, tj. umožňuje funkci aplikace na všech operačních systémech a jazycích[36].

2.11 ZeroMQ

ZeroMQ je vysoce výkonná asynchronní knihovna zpráv, zaměřena na použití v distribuovaných nebo paralelních aplikacích. Jeho hlavní důležitou vlastností, kterou se výrazně liší od předešlých protokolů, je že ke své funkcionalitě nepotřebuje broker server[37].

Tento projekt byl nastartován roku 2007 a později roku 2010 některé společnosti opustily pracovní skupinu zabývající se protokolem AMQP s úmyslem podporovat výrazně rychlejší a jednodušší ZeroMQ. ZeroMQ byl standardně vytvořen pro jazyk C++ a v roce 2012 byly vytvořeny konverze JeroMQ pro jazyk Java a NetMQ pro jazyk C#[37].

2.11.1 Specifikace

Mezi hlavní výhody ZeroMQ patří rychlost. ZeroMQ pro přenos zpráv využívá transportní protokol Multicast, který je účinný pro přenos dat na více míst určení. Webová stránka Second Life provedla průzkum a zveřejnila end-to-end latenci 13,4mikrosekund a až 4 100 000 zpráv za sekundu[38].

Jednoduchost tohoto protokolu spočívá v použití socketů. ZeroMQ umožňuje asynchronní I/O operace rozdělit do vláken a je tak schopen manipulovat síťový provoz a odvést tuto práci za nás[38].

ZeroMQ je na rozdíl od jiných tradičních systémů tzv. brokerless, tudíž systém neobsahuje centralizovaný broker server ve středu sítě, ke kterému by byl každý uzel připojen, a uzly by vzájemně komunikovaly mezi sebou pouze přes tento centrální server. V ZeroMQ je umožněno, aby aplikace komunikovaly mezi sebou, bez jakéhokoliv zprostředkovatele ve středu sítě[38].

2.11.2 Technologie

ZeroMQ vyžaduje, aby byly pro komunikaci použity optimalizované základní vzory a aby byly nekonečně škálovatelné[37]:

- Žádost-odpověď (Request-Reply) - jedná se o nejčastější a nejpoužívanější způsob použití ZeroMQ, spočívá ve volání vzdálených procedur a úlohu distribučního vzoru. Připojuje sadu klientů k souboru služeb. Musí být obdržena odpověď na každou odeslanou zprávu.
- Publikovat-odebírat (Publish-Subscribe) - jedná se o vzor jednosměrné distribuce dat, který připojuje vydavatele (Publish clienta) k sadě odběratelů (Subscribe clients). V podstatě server posílá zprávy sadě klientů a zprávy jsou přijímány pouze těmi klienty, kteří o ně stojí [38].
- Tlačit-táhnout (Push-Pull) volně překládáno jako potrubí, jedná se o paralelní úlohy distribuce a sběru dat, kdy jsou uzly propojeny pomocí fan-in a fan-out vzoru a data se přenáší mezi těmito uzly v potrubí a jsou předávány průběžně.
- Exkluzivní dvojice (Exclusive pair) - jedná se o vzor pro specifikované případy použití, spojuje dva sockety v jeden pár.

Každý vzor definuje konkrétní topologii sítě. Žádost-odpověď definuje topologii zvanou jako sběrnice (bus service), Publikovat-odebírat stanovuje strom pro distribuci dat (data distribution tree) a vzor Tlačit-táhnout definuje paralelní potrubí (parallelised pipeline)[37].

3 Konfigurace klientů a serveru

Klíčovým prvkem komunikace je použití vhodných softwarových kombinací, které budou splňovat požadavky na funkčnost, jak na straně klienta, tak na straně serveru. Respektive zajistí korektní odesílání a přijímání zpráv na obou stranách, neboť každý klient může být odesílatelem i příjemcem. Důležitá je správná kombinace vybraného komunikačního protokolu vůči Broker serveru, který slouží jako prostředník mezi klienty a zprostředkovává komunikaci a koordinuje provoz zpráv.

3.1 Použitá zařízení a schéma zapojení

Všechny testy byly provedeny na notebooku v roli broker serveru s těmito parametry:

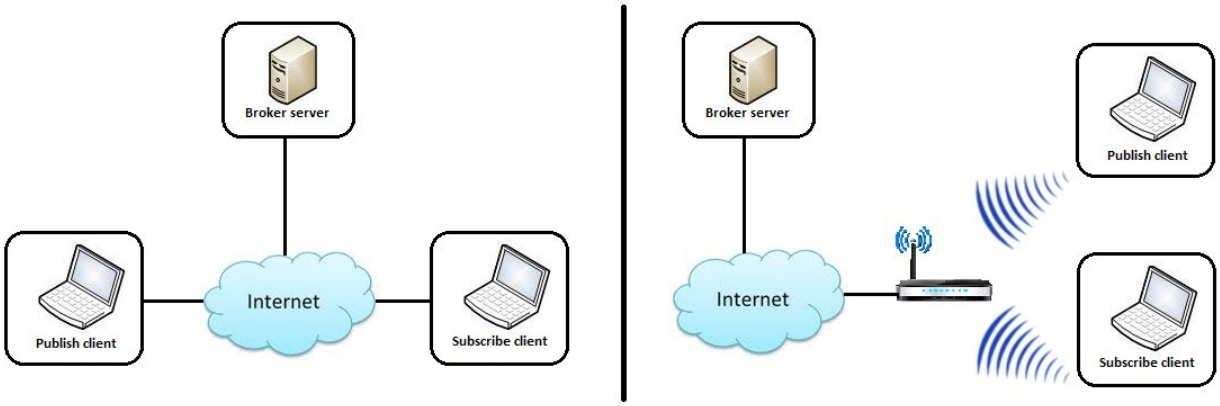
- Typ: Lenovo E531
- OS: MS Windows 8
- RAM: 4GB
- CPU: Intel Core i5

Všechny testy byly provedeny na zařízeních v roli klienta s těmito parametry:

- Klient č.1 Notebook: MSI GP60 2QF-1092XCZ
- OS: MS Windows 10
- RAM: 4GB
- CPU: Intel Core i5
- Klient č.2 Notebook: ASUS ZENBOOK UX303UB R41015T
- OS: MS Windows 8
- RAM: 8GB
- CPU: Intel Core i5

3.1.1 Schéma zapojení

Níže uvedené schéma zapojení se vztahuje ke všem vybraným testovaným protokolům, kromě protokolu ZeroMQ, jehož schéma je uvedeno zvlášť v sekci ZeroMQ, a sice v poslední části této kapitoly, neboť tento protokol jako jediný neobsahuje broker server.



Obrázek 3.1: *Schéma zapojení* pro varianty testování A a B

3.2 MQTT

Jelikož se jedná o velmi rozšířený a oblíbený protokol, existuje mnoho možností při výběru broker severu. K nejčastějším a nejznámějším patří Mosquitto, HiveMQ, ActiveMQ, RabbitMQ aj. Nicméně, z důvodu nalezení komplexně zpracované knihovny M2Mqtt pro klienta, jsem se rozhodla použít pro svůj účel GnatMQ broker server zpracovaný a dostupný pro všechny platformy pod záštitou .NET. Vše lze nalézt v přílohách.

Tyto komplexně zpracované knihovny obsahují MQTT klienta, schopného se připojit k MQTT broker serveru. K jejich zprovoznění však bylo nutné doprogramovat určité části kódu, zásadní z nich budou popsány níže. Jsou vytvořeny v jazyce C# a jsou podporovány platformou OS Windows 8 a vyšší, což splňuje moje požadavky na systém. Také pracují pod platformou .NET Framework 4.5 a vyšší, tudíž k implementaci knihoven je Visual Studio 2013 plně dostačující.

3.2.1 Implementace

Nejprve bylo nutné provést zapojení zařízení podle schématu, poté je možné začít s konfigurací GnatMQ broker serveru. Server jsem spustila s využitím knihoven MqttBroker serveru: GnatMQ. Server bylo snadné vytvořit a spustit, neboť knihovna má srozumitelné a výstižné API. Vyvolání broker serveru proběhlo jen za pomocí tohoto kódu:

```
MqttBroker broker = new MqttBroker();

broker.Start();
```

Dále do třídy M2MqttClientSender definuji konstantní proměnné pro Quality of Services všech tří úrovní MQTT.

```
public const byte QOS_LEVEL_AT_MOST_ONCE = 0x00;

public const byte QOS_LEVEL_AT_LEAST_ONCE = 0x01;

public const byte QOS_LEVEL_EXACTLY_ONCE = 0x02;
```

V hlavní metodě Main třídy ClientSender inicializuji klienta. Je nutné mu přiřadit IP adresu serveru, ke kterému se chce klient připojit včetně portu, přes který bude komunikovat. Připojení probíhá pomocí metody Connect k broker serveru, kde se klient prezentuje jako Publisher. To stejné platí i pro Subscribe klienta.

```
Console.Write("Enter the IP address of the server [a.b.c.d |
a.b.c.d:port]: ");

string IP = Console.ReadLine();

MqttClient client = new MqttClient(IP);

client.Connect("publisher");

Console.WriteLine("Connected: " + client.IsConnected);
```

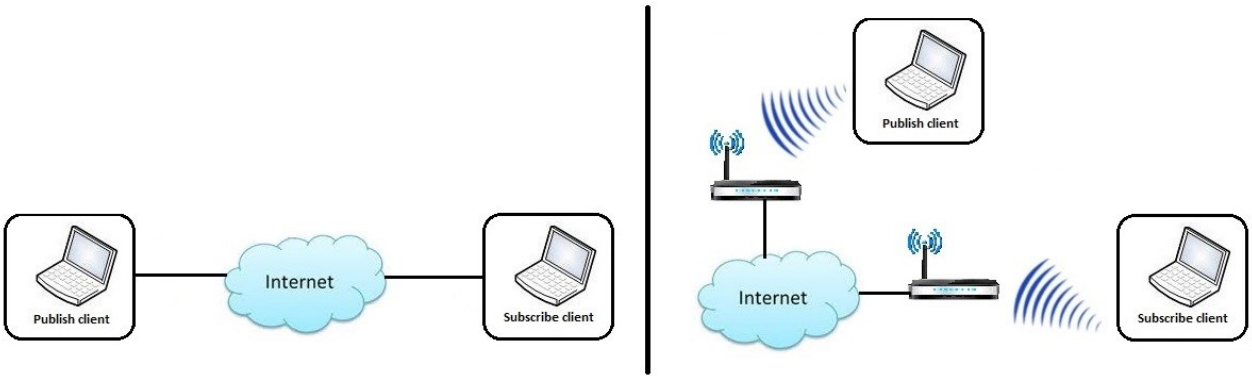
Dále se v této třídě vyskytuje statická metoda Send, jejíž účelem je vypsat do souboru "publisher.txt" čas odeslání zprávy ve formátu DateTime s přesností na milisekundy, včetně její délky, resp. velikosti obsahu zprávy a použitého QoS. Na stejném principu pracuje metoda Client_MqttMsgPublishReceived, která zapisuje do souboru "subscriber.txt" data o přijetí zprávy.

3.3 ZeroMQ

Vzhledem k nalezení velmi hezky a přehledně zpracované dokumentace k jednoduchému použití a implementaci ZeroMQ protokolu, viz zde [39], jsem se rozhodla implementovat tento protokol prostřednictvím jazyka Java, neboť dokumentace obsahuje ukázky implementace v tomto jazyce.

K implementaci byl použit nástroj Apache Maven, který poskytuje repository obsahující všechny javové knihovny a odpadá tak nutnost, starat se o všechny knihovny a jejich požadavky. Bylo tedy využito konverze ZeroMQ použitím knihovny JeroMQ pro jazyk Java. Další výhodou tohoto nástroje je, že umožňuje jediným příkazem zabalit výsledek projektu do spustitelného souboru JAR a spustit jej tak na jakémkoli jiném počítači, což se uplatnilo zejména při testování.

Protokol ZeroMQ je na rozdíl od AMQP a MQTT protokolu brokerless, a proto má odlišné schéma zapojení, které je zobrazeno níže.



Obrázek 3.2: Schéma zapojení ZeroMQ protokolu pro obě varianty testování

3.3.1 Implementace

Po zapojení zařízení dle schématu, bylo možné začít s konfigurací klienta v roli serveru. Pro tyto účely testování byl použit vzor Request-Reply, jelikož je dostačující a jedná se o nejoblíbenější a nejpoužívanější vzor pro snadnou implementaci ZeroMQ. V prvé řadě bylo nutné implementovat samotnou knihovnu pro ZeroMQ.

```
import static org.zeromq.ZMQ.*;
```

Poté bylo možné pokračovat s implementací statických proměnných, do kterých byl přiřazen port 5555 pro TCP spojení, dále formát aktuálního data a času včetně milisekund, dále název souboru "server.csv" do kterého se mají ukládat informace o přijetí zprávy pomocí BufferedWriter.

```
private static final String SERVER_URL = "tcp://*:5555";

private static final DateTimeFormatter DTF =
DateTimeFormatter.ofPattern("dd.MM.yyyy HH:mm:ss,SSS");

private static final String LOG_FILE = "server.csv";

private static BufferedWriter writer;
```

Dále je v hlavní metodě Main obsaženo nachystání kontextu a socketu s odpovědí žadateli tj. Publish klientovi, je zinicializován soubor a dále je obsažen stěžejní cyklus while, jehož úkolem je získat

a zapsat parametry přijetí zpráv do souboru dokud je vlákno aktivní a informovat druhou stranu o úspěšném přijetí. V opačném případě, je komunikace ukončena příkazem "responder.close();".

```
Context context = context(1);
Socket responder = context.socket(REP);
inicializeFile();
responder.bind(SERVER_URL);
while (!Thread.currentThread().isInterrupted()) {
String request = responder.recvStr();
LocalDateTime now = LocalDateTime.now();
if (KILLYOURSELF.equalsIgnoreCase(request)) {
break; }
writeToFile(now, request.length());
responder.send("true".getBytes(), 0); }
```

Dále tato třída Server obsahuje metodu createMessage, která vrací typ byte, metodu inicializeFile, která má za úkol vytvořit a otevřít soubor s nadepsanou hlavičkou, metodu writeToFile, která zapisuje do souboru informace o přijetí zprávy a metodu closeFile, která soubor uzavírá.

Třída Client je velmi podobná třídě Server. Obsahuje také výše zmíněné čtyři metody, včetně téhož importu knihovny ZeroMQ. Obsahuje téměř stejné statické proměnné jako Server, stejný TCP port 5555, pouze název vytvářeného souboru "client.csv" se liší. Třída Client obsahuje navíc pouze vygenerování listu/seznamu zpráv a připojení klienta přes TCP port, přes který bude komunikovat.

```
List<Integer> MESSAGES = Arrays.asList(1000, 5000, 10000, 25000,
50000, 80000, 100000, 250000, 500000, 1000000);
requester.connect(CLIENT_URL);
```

Tento list/seznam zpráv je jednotlivě procházen a na základě určené velikosti hodnoty zprávy jsou vytvářeny zprávy pomocí metody createMessage, které jsou odesílány protistraně. Zároveň je jak na obrazovku, tak i do souboru vypsána informace (čas, velikost) o dané zprávě.

```
MESSAGES.forEach(m -> {
LocalDateTime now;
            for (int i = 0; i < 10; i++) {
                now = LocalDateTime.now();
                System.err.println(System.currentTimeMillis());
                requester.send(createMessage(m), m);
                String x = writeToFile(now, m);
                requester.recv();
            }   });
```

Po ukončení cyklu a odeslání všech zpráv, žadatel/klient/publisher posílá informaci o ukončení komunikace, ukončuje spojení a zavírá soubor.

```
requester.send(KILLYOURSELF);

requester.disconnect(CLIENT_URL);

requester.close();

closeFile();
```

Po dokončení implementace byl výsledný finální zdrojový kód zabalen do spustitelného JAR souboru, jak pro Server, tak i pro Clienta. Tyto soubory i zdrojové kódy lze nalézt v přílohách, včetně naměřených a zpracovaných hodnot. JAR soubory byly spuštěny z příkazového řádku ve formátu pro server:

```
java -jar C:\Users\Nikol\Desktop\ZeroMQServer-1.0-jar-with-
dependencies.jar
```

A pro klienta ve stejném formátu s přidanou IP adresou serveru, jak je uvedeno zde:

```
java -jar C:\Users\Nikol\Desktop\ ZeroMQ-1.0-jar-with-
dependencies.jar 10.0.0.8
```

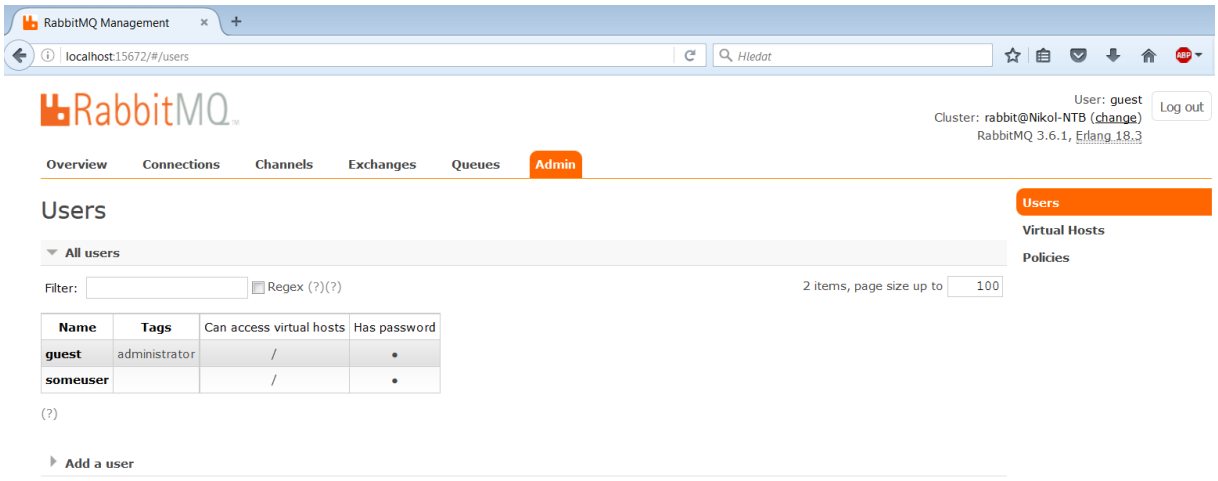
Aby navázání spojení proběhlo a komunikace probíhala úspěšně, je nutné povolit protokol ZeroMQ a jemu příslušný port, přes který komunikace probíhá tj. 5555, v opačném případě by bylo nutné vypnout veškeré antiviry a firewally, které by blokovaly její průběh, avšak to se z bezpečnostního hlediska nedoporučuje.

3.4 AMQP

K implementaci tohoto protokolu jsem využila známého broker serveru RabbitMQ, jehož přehlednou a snadno pochopitelnou dokumentaci lze najít k vidění zde [40]. Jelikož se jedná o dokumentaci psanou pro programovací jazyk Java, rozhodla jsem se pro implementaci klienta a serveru prostřednictvím jazyka Java. K implementaci byl použit nástroj Apache Maven, který poskytuje repository obsahující všechny javové knihovny a odpadá tak nutnost, starat se o všechny knihovny a jejich požadavky.

3.4.1 Implementace

Nejprve bylo nutné vytvořit implementaci broker serveru, počínaje nainstalováním 64bitové verze multiprogramovacího jazyka Erlang, bez kterého by RabbitMQ server nebyl schopen pracovat. Po dokončení instalace bylo dále potřebné stažení a instalace samotného RabbitMQ broker serveru verze 3.6.1., kde byl dále spuštěn a nainstalován pomocí příkazové řádky plugin "rabbitmq_management" pro administraci RabbitMQ serveru. Po úspěšném dokončení všech instalací bylo možné pracovat pohodlně v samotném GUI pro RabbitMQ, kde bylo po přihlášení s uživatelským jménem "guest" a heslem "guest" nutné vytvořit sdíleného klienta a přidělit mu uživatelské jméno, heslo a uživatelská práva virtuálního hosta, viz zmíněný obrázek. Tedy v mém případě se jednalo o uživatele "someuser" s přístupovým heslem "1234". Dále se veškerá implementace odehrávala ve vývojovém prostředá Apache Meaven, tedy implementace Publish klienta a Subscribe klienta.



Obrázek 3.3: *Vytvoření a přidání* uživatelského účtu v Administrátorském prostředí RabbitMQ broker serveru

Po zapojení konfigurace dle schématu a instalace RabbitMQ broker serveru, mohla následovat implementace obou klientů v jazyce Java, která začala importem potřebné knihovny pro komunikaci se serverem.

```
import com.rabbitmq.client.*;
```

Implementace si je na první pohled velmi podobná s implementací protokolu ZeroMQ. Jak pro příjemce, tak i pro klienta odesílajícího zprávy, jsou vytvořeny základní statické proměnné pro uložení formátu data a času včetně milisekund formou DateTime, dále soubor Consumer.csv nebo Producer.csv, do kterého jsou zapisovány informace o přijaté/odeslané zprávě pomocí BufferedWriter. Navíc je deklarována pouze fronta zpráv pro AMQP protokol tak, aby zprávy byly přijímány v takovém pořadí, jako byly odesílány a v případě Producer/Publisher klienta je navíc vytvořen seznam/list definující velikost odesílané zprávy.

```
private static final String LOG_FILE = "producer.csv";

private static BufferedWriter writer;

private static final DateTimeFormatter DTF =
DateTimeFormatter.ofPattern("dd.MM.yyyy HH:mm:ss,SSS");

private static final List<Integer> MESSAGES = Arrays.asList(1000,
5000, 10000, 25000, 50000, 80000, 100000, 250000, 500000, 1000000);

private final static String QUEUE_NAME = "amqp";
```

Dále je vytvořeno spojení mezi klientem a broker serverem přes takzvaný komunikační kanál, jak lze vidět níže, kde se klient přihlašuje k broker serveru přes svoje uživatelské jméno a přes své přístupové heslo, které jsme již dříve nastavili v Administrátorském prostředí RabbitMQ broker serveru. V Administrátorském prostředí je možné vytvořit více uživatelských účtů, respektive zvlášť účet pro

Consumer/Subscribe klienta a zvlášť pro Producer/Publisher klienta, z důvodu jednoduchosti jsem použila jeden sdílený účet. Nakonec je přidána i URL adresa serveru, ke kterému se přihlašujeme.

```
ConnectionFactory factory = new ConnectionFactory();
factory.setUsername("someuser");
factory.setPassword("1234");
factory.setHost(SERVER_URL);
Connection connection = factory.newConnection();
Channel channel = connection.createChannel();
```

V komunikačním kanálu je řízená fronta, jakmile jsou zprávy na straně Publishera odeslány, je kanál, spojení i soubor do kterého byly uloženy informace o odeslaných zprávách, uzavřen. Stejný princip platí i pro příjemce, který po přijetí všech zpráv a zapsání všech údajů o zprávách do souboru, taktéž uzavře spojení.

```
channel.queueDeclare(QUEUE_NAME, false, false, false, null);
sendMessages(channel, prefatch);
channel.close();
connection.close();
closeFile();
```

Stejně jako protokol ZeroMQ, obsahuje i AMQP shodné metody pro odesílání zpráv pomocí metody sendMessages, kdy však v parametru metody musí být určeno, přes který komunikační kanál zpráva bude odeslána a přes jaký typ QoS, protože na rozdíl od ZeroMQ, AMQP poskytuje tři typy Quality of Services, stejně jako MQTT protokol. Určit, přes jaký typ QoS si přejeme zprávy posílat, můžeme při spuštění spustitelného JAR souboru jednoduchým zadáním parametru 0,1 nebo 2, které označují úrovně QoS tak, jak jsou definovány. Implementace obsahuje také metody createMessage, inicializeFile, writeToFile, closeFile, které byly už popsány výše.

Po dokončení implementace byl výsledný finální zdrojový kód zabalen do spustitelného JAR souboru, jak pro Consumer/Subscriber clienta, tak i pro Producer/Publisher clienta. Tyto soubory i zdrojové kódy lze nalézt v přílohách, včetně naměřených a zpracovaných hodnot protokolu AMQP. Aby navázání spojení proběhlo a komunikace probíhala úspěšně, je nutné povolit protokol AMQP a jemu příslušný port, přes který komunikace probíhá, v opačném případě by bylo nutné vypnout veškeré antiviry a firewally, které by blokovaly její průběh, avšak to se z bezpečnostního hlediska nedoporučuje. JAR soubory byly spuštěny z příkazového řádku v tomto formátu s přidanou IP adresou zařízení, na kterém se nachází broker server a s přidaným parametrem QoS, viz zde:

```
java -jar C:\Users\Nikol\Desktop\AMQPConsumer-1.0-jar-with-
dependencies.jar 10.0.0.2 1
```

```
java -jar C:\Users\Nikol\Desktop\AMQPProducer-1.0-jar-with-
dependencies.jar 10.0.0.2 1
```

4 Výsledky testování vybraných protokolů

Z výkonnostního hlediska jsem se při testování zaměřila především na testování rychlosti vybraných protokolů a s tím spojeným zpožděním, které neblaze ovlivňuje kvalitu komunikace, a ke kterému dochází při konkrétní implementaci jednotlivých protokolů.

Cílem tohoto testování bylo zasílání určitého množství zpráv o různé velikosti obsahu z prvního klienta na druhého přes broker server. Testování probíhalo s velikostně různými hodnotami, nicméně se během testování ukázalo, že níže uvedené hodnoty jsou k zachycení výsledků dostačující a z hlediska grafického zpracování výsledků nejlépe čitelné a pro tento účel, tedy ideální. V případě nižších velikostí obsahu zprávy, byl rozdíl jednotlivých výsledků testů zanedbatelný a naopak při mnohem větších velikostech zpráv byl rozdíl markantní příliš a docházelo k velké zátěži sítě a časovým prodlevám. Postupně proto byly zasílány zprávy o velikosti:

- 1kB
- 5kB
- 10kB
- 25kB
- 50kB
- 80kB
- 100kB
- 250kB
- 500kB
- 1MB

Po přeložení a odladění kódu, bylo možné provést testování protokolu ve formě zasílání zpráv různé velikosti obsahu, tzn., bylo odesláno 10 typů zpráv. Toto testování bylo provedeno pro každou z úrovní QoS. Toto měření bylo uskutečněno 10krát. Data, konkrétně časový rozdíl mezi odesláním zprávy a jejím přijetím, byly vloženy do MS Office Excel 2013, kde byl z časových rozdílů spočítán průměrný čas putování konkrétní zprávy pro konkrétní QoS. Tyto výsledky měření byly následně graficky zpracovány tak, aby bylo zpoždění mezi koncovými zařízeními patrné. Pro každý typ testu, byl tedy výsledný jeden graf tvořen z 300 hodnot.

Testování bylo vždy uskutečněno pro dvě varianty přenosového média, a sice:

- Test A: Přenos dat probíhal přes Fast Ethernet s přenosovou rychlostí 100Mbit/s definovaný standardem IEEE 802.3u
- Test B: Bezdrátový přenos dat přes Wi-Fi standart IEEE 802.11g, jenž pracuje v přenosovém pásmu 2,4GHz s přenosovou rychlostí 54Mbit/s

Poslední typ testu spočíval ve snaze maximálního zatížení přenosového média, v mém případě se jednalo o zatížení bezdrátové sítě, dalšími účastníky sítě, kteří síť vytížili stahováním filmů a streamováním videa. V takto zhoršených podmínkách bylo znova uskutečněno testování, které má ukázat, který z protokolů si v kritických podmínkách povede nejlépe. Tento test je označován jako:

- Test C: Přenos dat v zatížené bezdrátové síti

Zde je uveden příklad zdrojového kódu z pozice odesílatele, resp. Publish klienta, který vygeneruje ze třídy Random zprávy, naplněné náhodnými hodnotami, o velikosti obsahu 1kB, 5kB, 10kB, 25kB, 50kB, 80kB, 100kB, 250kB, 500kB a 1MB a následně uživatele informuje o tom, že zprávy jsou vygenerovány a čeká se pokyn k odeslání (enter).

```
Random r = new Random();
byte[] mesByte1K   = new byte[1000];
byte[] mesByte5K   = new byte[5000];
byte[] mesByte10K  = new byte[10000];
byte[] mesByte25K  = new byte[25000];
byte[] mesByte50K  = new byte[50000];
byte[] mesByte80K  = new byte[80000];
byte[] mesByte100K = new byte[100000];
byte[] mesByte250K = new byte[250000];
byte[] mesByte500K = new byte[500000];
byte[] mesByte1M   = new byte[1000000];
r.NextBytes(mesByte1K);
r.NextBytes(mesByte5K);
r.NextBytes(mesByte10K);
r.NextBytes(mesByte25K);
r.NextBytes(mesByte50K);
r.NextBytes(mesByte80K);
r.NextBytes(mesByte100K);
r.NextBytes(mesByte250K);
r.NextBytes(mesByte500K);
r.NextBytes(mesByte1M);
Console.WriteLine("Random values are generated.");
Console.WriteLine("Press enter to start");
```

Dále použitím třídy StreamWriter a metody Send, lze vidět, že zprávy jsou jednotlivě odesílány Publish klientem s přiřazeným parametrem QoS, dle úrovně kterou požadujeme. Pro příklad mám uvedenou QoS úrovně 0. Parametry metody Send definují, o jakou zprávu se jedná a počet, kolikrát má být odeslána, čili v mém případě 10krát. Do souboru "publisher.txt" definovaném hned na začátku, jsou vždy vypsány informace ohledně zprávy, pro každou konkrétní zprávu zvlášť tak, aby bylo posléze možné zjistit čas putování zprávy od zdroje k cíli a určit za jakých okolností a k jakému zpoždění dochází. Tento příklad se konkrétně vztahuje k MQTT protokolu.

```
string fileName = "publisher.txt";
```

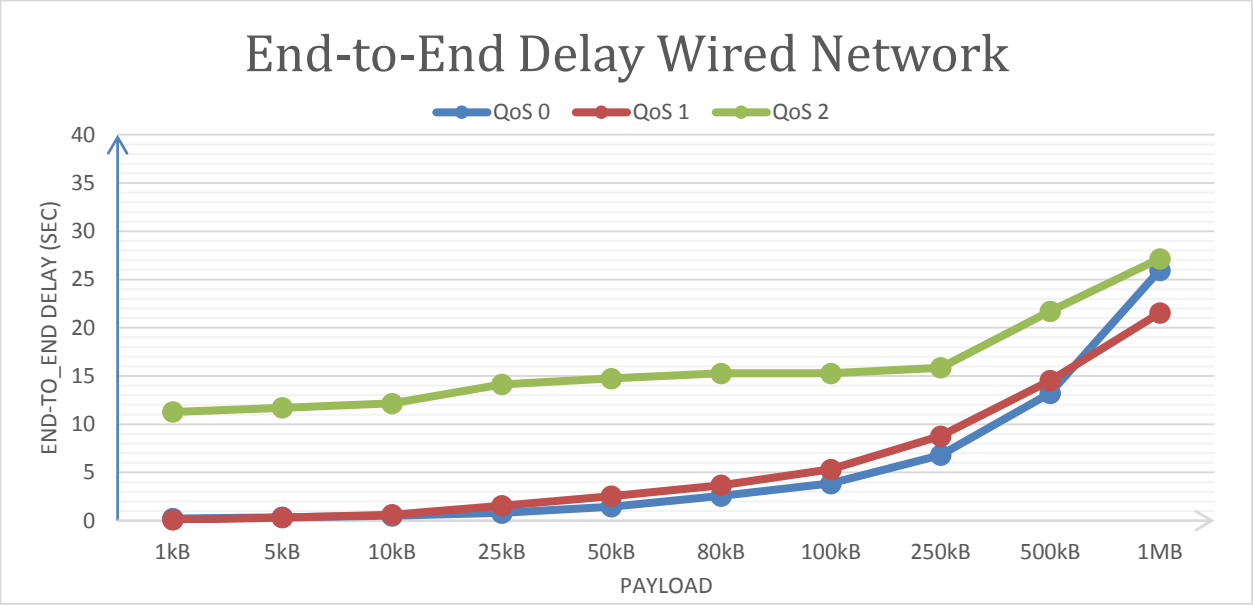
```
using (StreamWriter file = new StreamWriter(fileName, false,
Encoding.UTF8))
Console.WriteLine("Sending QOS_LEVEL_AT_MOST_ONCE");
Send(client, file, qos[0], mesByte1K, 10);
Send(client, file, qos[0], mesByte5K, 10);
Send(client, file, qos[0], mesByte10K, 10);
Send(client, file, qos[0], mesByte25K, 10);
Send(client, file, qos[0], mesByte50K, 10);
Send(client, file, qos[0], mesByte80K, 10);
Send(client, file, qos[0], mesByte100K, 10);
Send(client, file, qos[0], mesByte250K, 10);
Send(client, file, qos[0], mesByte500K, 10);
Send(client, file, qos[0], mesByte1M, 10);
```

Níže je uvedeno jakou úlohu metoda Send zastává a jaké vstupní parametry obsahuje. Pokud je splněna vstupní podmínka, že se jedná o jednu z úrovní QoS, může být přistoupeno k zápisu do souboru, kde pomocí cyklu for jsou do souboru zapsány informace pro konkrétní typ zprávy tolikrát, kolikrát je stanoven počet jejího odeslání. Jedná se o čas odeslání zprávy ve formátu DateTime s přesností na milisekundy, včetně velikosti zprávy a použitého QoS.

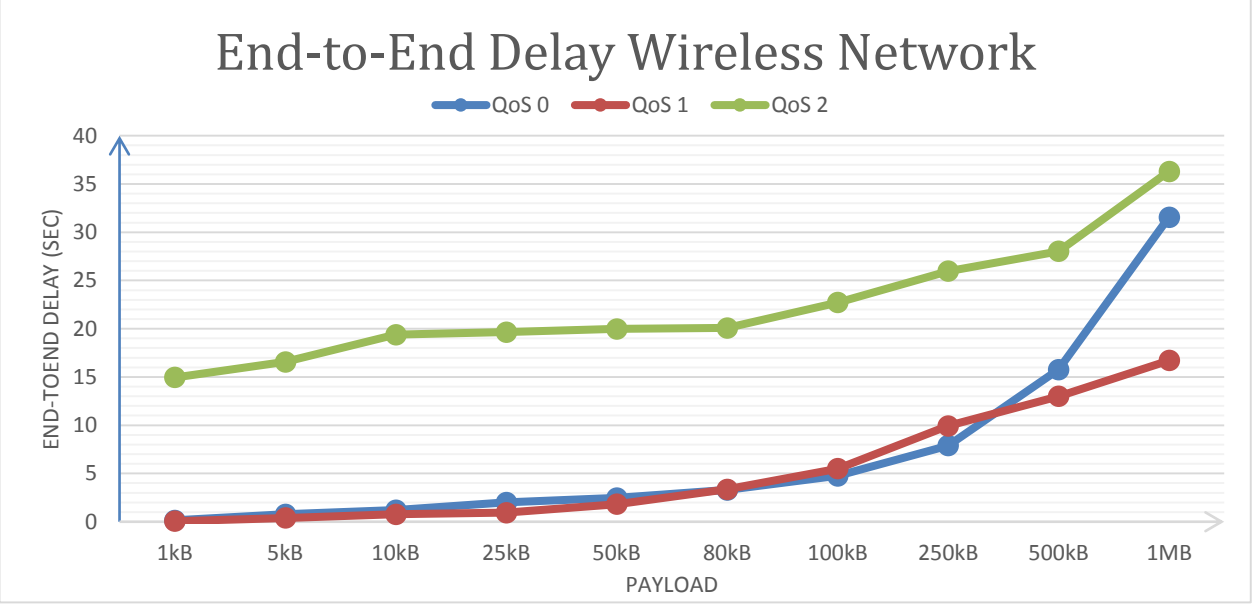
```
private static void Send(MqttClient client, StreamWriter file, byte
QoS, byte[] array, int count)
{
            string topic = "topic1";
            if (QoS.Equals(QOS_LEVEL_AT_MOST_ONCE))
                topic = "topic1";
            else if (QoS.Equals(QOS_LEVEL_AT_LEAST_ONCE))
                topic = "topic2";
            else if (QoS.Equals(QOS_LEVEL_EXACTLY_ONCE))
                topic = "topic3";

            for (int i = 0; i < count; i++){
                array[0] = QoS;
                array[1] = (byte)i;
file.WriteLine(DateTime.Now.ToString("0:dd.MM.yyyy HH:mm:ss.fff") +
";" + array.Length + ";" + QoS+";"+(byte)i);
                client.Publish(topic, array, QoS, false);}}
```

4.1 MQTT

Graf uvedený níže, znázorňuje analýzu zachyceného přenosu dat přes Fast Ethernet pro všechny tři úrovně QoS protokolu MQTT. Z grafu je patrné, že výsledky zpoždění zpráv se pro QoS-0 a QoS-1 moc neliší, zvláště pro zprávy do 25kB je jejich zpoždění téměř totožné a zanedbatelné. Od hodnoty 50kB zpoždění pro obě QoS mírně narůstá až po hodnotu 200kB, kde začíná prudce stoupat až k 25sec. Zpoždění zpráv zasílaných prostřednictvím QoS-2 začíná na 11sec a ukazuje se jako stabilní až po zprávy o velikosti 250kB, kde stejně jako ostatní QoS začíná prudce stoupat. Nejlépe, si tak v tomto testu vede QoS-0, což splňuje předpoklady, neboť zprávy jsou odesílány a nijak není potvrzováno jejich doručení, dochází k nejmenšímu zpoždění. Směle mu může konkurovat QoS-1, které má srovnatelné výsledky a je jistě spolehlivější. Naopak, dle předpokladů se u QoS-2 projevilo největší zpoždění, které je zapříčiněno právě 4-way handshake, které dělá toto QoS spolehlivým.

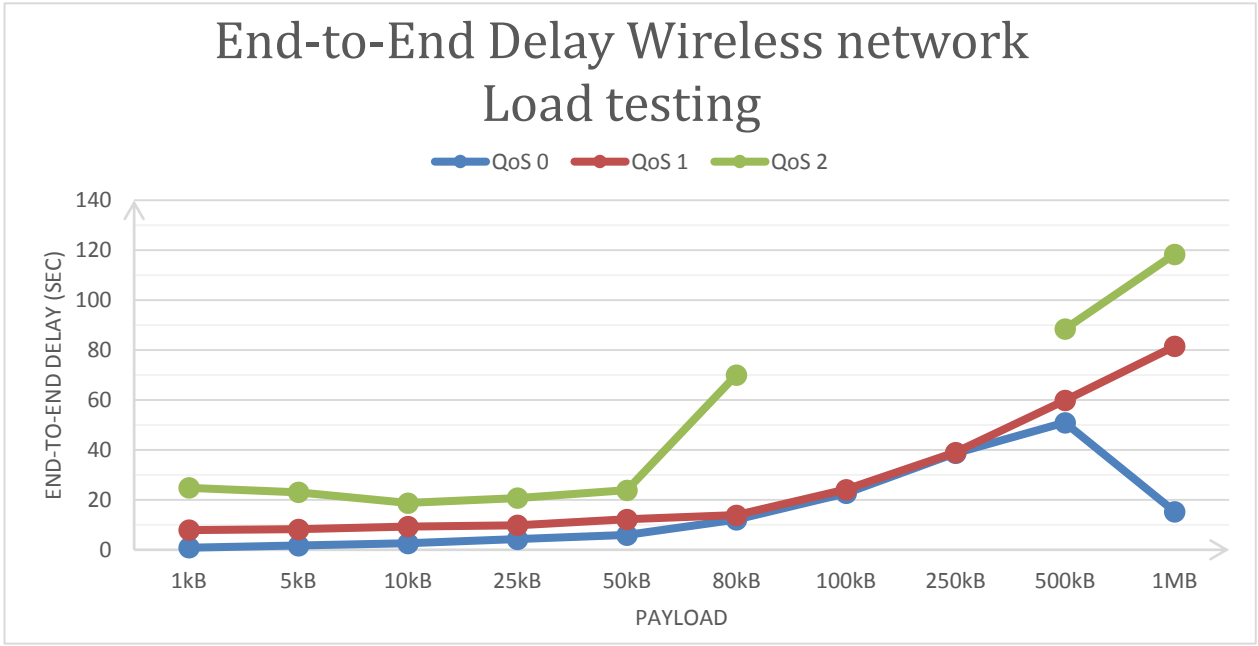


Graf uvedený níže, znázorňuje analýzu zachyceného provozu pro bezdrátový přenos pro všechny tři úrovně QoS protokolu MQTT. Rozbor grafu je popsán na další straně.



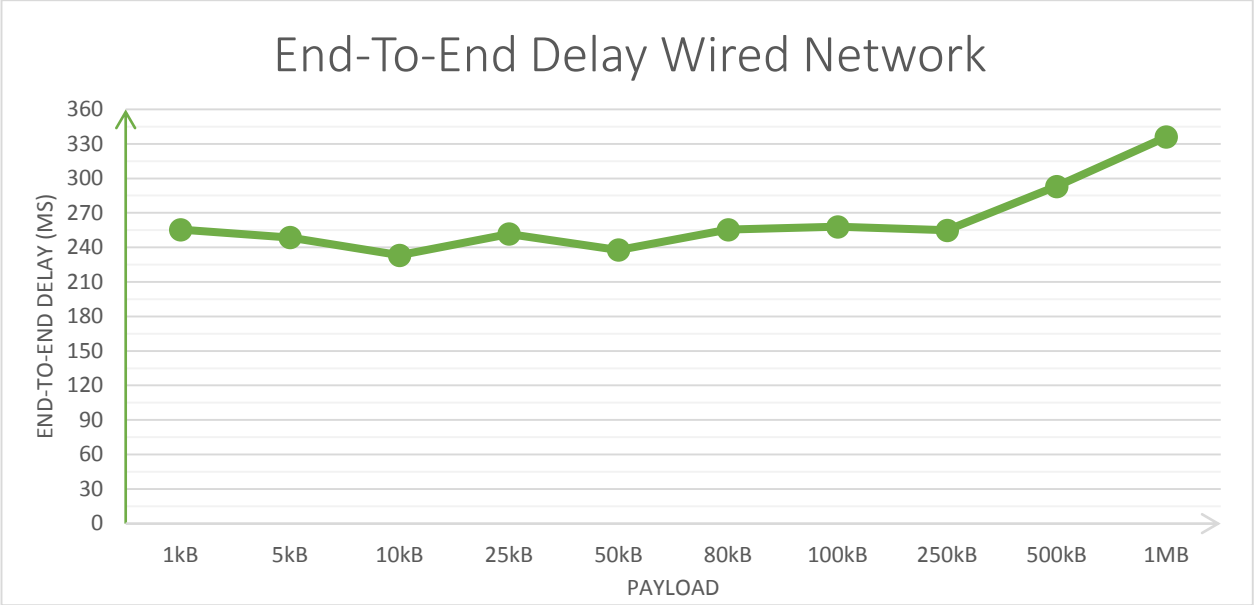
Měření zpoždění pro bezdrátový přenos se ukázalo být pro QoS-0 a QoS-1 překvapivě vyrovnané a srovnatelné s výsledky pro Fast Ethernet přenos. Stejně jako prve, zpoždění začíná na zanedbatelných hodnotách a až od zpráv velkých 80kB dochází k nárůstu zpoždění pro obě QoS 0 i 1. Nicméně koncové zpoždění QoS-0 pro zprávy 1MB, které odpovídá hodnotě přibližně 31sec nepovažuji za relevantní, domnívám se, že došlo k momentálnímu zatížení sítě, které se neblaze projevilo na měření. Naopak pro QoS-2 jsou výsledky bezdrátového přenosu viditelně horší než u přenosu drátového. Zpoždění začíná už na 15sec, což je o 5sec horší než prve a postupně roste až na přibližně 36sec pro zprávy o velikosti 1MB, což je ve srovnáním s drátovým přenosem až o 8sec větší zpoždění. Z výsledků vyplývá, že na bezdrátovém přenosu bude výsledné zpoždění vždy větší, což je zapříčiněno především nižší přenosovou rychlostí a half-duplex provozem.

Poslední níže uvedený graf pro MQTT protokol, se vztahuje k testu C, který jak již bylo zmíněno, spočívá v zachycení vlivu zatížení bezdrátové sítě na přenos zpráv. S jistotou lze říci, že dle předpokladů došlo ke zvýšení zpoždění. Nejméně zatížení sítě zasáhlo QoS-0, jehož zpoždění začínalo na necelé 1sec, a které až do 50kB vzrostlo pouze dvojnásobně ve srovnání s nezatíženou wireless sítí, poté se hodnoty zpoždění začaly zvyšovat až čtyřnásobně. Nejkritičtějším bodem byl přenos 500kB zpráv, kde došlo ke zpoždění téměř 51sec, poté došlo k prudkému poklesu, z čehož lze předpokládat, že zatížení sítě se nečekaně snížilo. Mnohem hůře si v testu vedl QoS-1, kde zpoždění začínalo už přibližně na 9sec a do 80kB vzrostlo až desetinásobně ve srovnání s nezatíženou sítí. Poté se zpoždění stabilizovalo a pro větší zprávy bylo pouze 4krát větší než za obvyklého provozu. Kritickým bodem však pro QoS-1 byl přenos zpráv 1MB, jehož zpoždění trvalo 81sec, což je o 65sec více než v běžném provozu. Nejhůře v testu dopadlo QoS-2, které sice začínalo se zpožděním téměř 25sec, což není ani dvojnásobně větší zpoždění ve srovnáním s přenosem v nezatížené síti. Tuto vlastnost si dokázalo udržet až po zprávy o velikosti 50kB. Velký zlom nastal u zprávy 80kB, kde došlo ke zpoždění 70sec a začaly se ztrácet zprávy a poté dokonce došlo ke ztrátě všech zpráv/paketů o velikosti 100kB a 250kB a části zpráv o velikosti 500kB. Zprávy 1MB byly přijaty všechny v pořádku, ale s neuvěřitelným zpožděním téměř 120sec.

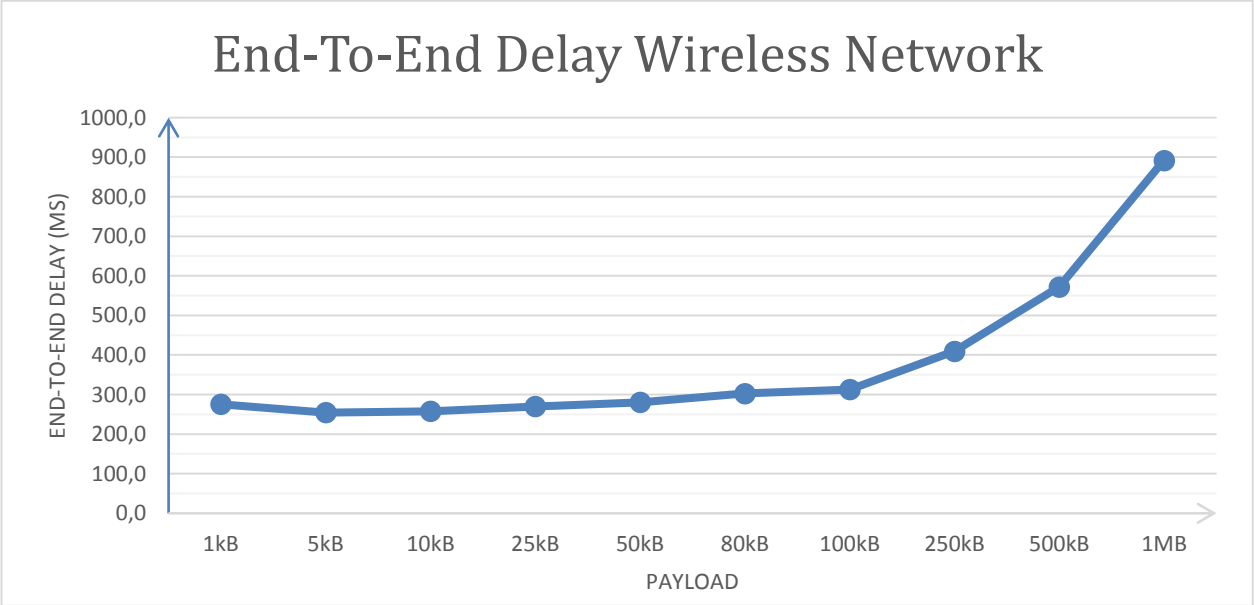


4.2 ZeroMQ

Graf uvedený níže, znázorňuje analýzu zachyceného přenosu dat přes Fast Ethernet pro protokol ZeroMQ. Upozorňuji, že výsledky jsou na rozdíl od MQTT protokolu uvedeny v milisekundách, což dokazuje, že přenos pomocí tohoto protokolu je opravdu rychlý. Zpoždění nejmenší zprávy 1kB začíná na 255ms a dá se říct, že až do odeslání zpráv o velikosti 250kB se nijak výrazně nemění a pohybuje se v rozmezí od 233ms do 257ms. K nárůstu dochází pří odesílání zpráv o velikosti 500kB a výše, kdy lze vidět, že křivka grafu se prudce zvedá až na hodnotu 335ms, což je i přes to stále velmi rychlé ve srovnání s MQTT protokolem, který na těchto hodnotách v podstatě začínal. Závěrem můžeme říci, že ZeroMQ se v tomto měření ukázal jako velmi stabilní protokol, minimálně do hranice 250kB.

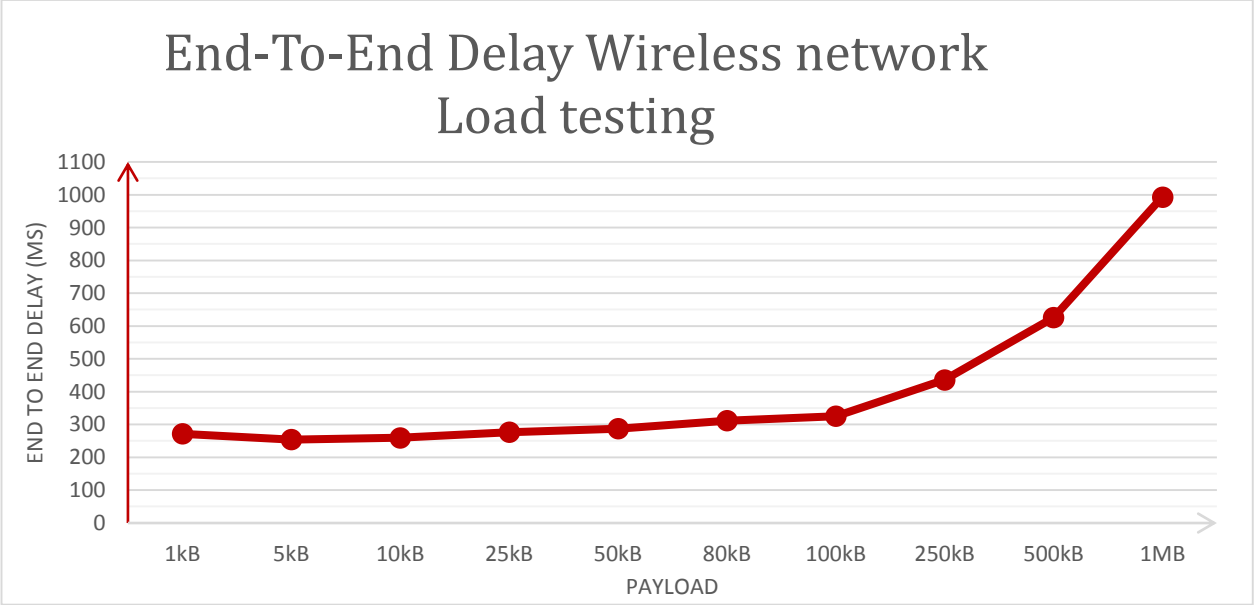


Graf uvedený níže, znázorňuje analýzu zachyceného provozu pro bezdrátový přenos pro protokol ZeroMQ. Rozbor grafu je popsán na další straně.



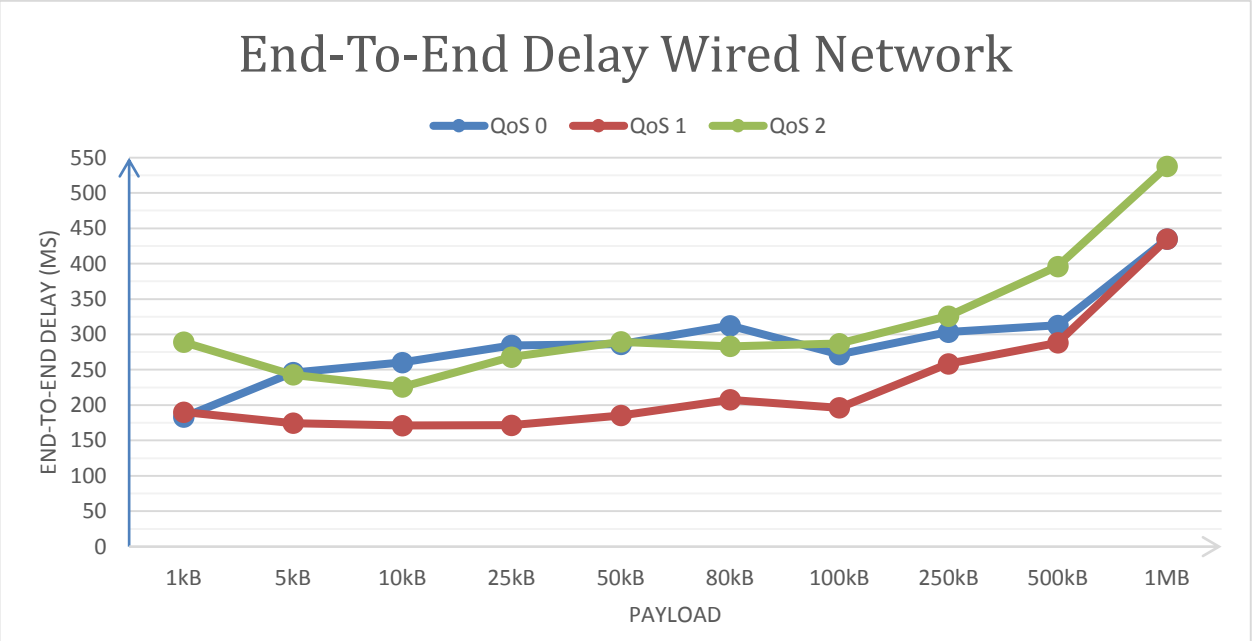
Z grafu je patrné, že zpoždění přenosu při nejmenších zprávách začíná na 275ms, což je pouze o 20ms více ve srovnáním s Fast Ethernet přenosem. Zpoždění zpráv od 1kB do velikosti zpráv 100kB se nijak výrazně nemění, je zpozorován maximálně 30ms nárůst. Tudíž můžeme opět říci, že se ZeroMQ do této hranice 100kB chová velmi stabilně. Avšak u zpráv o velikosti 250kB nastává zlom a křivka grafu prudce stoupá až na 900ms, což je téměř trojnásobek ve srovnáním s počátečními hodnotami zpoždění a také Fast Ethernet přenosem za stejných podmínek.

Poslední níže uvedený graf pro ZeroMQ protokol, se vztahuje k testu C, který jak již bylo zmíněno, spočívá v zachycení vlivu zatížení bezdrátové sítě na přenos zpráv. Výsledek testu dopadl velmi překvapivě, zdá se, že ani velké zatížení bezdrátové sítě nedělá protokolu ZeroMQ potíže. Hodnoty zpoždění sice dle předpokladů vzrostly, ale nijak závratně. ZeroMQ opět začíná se zpožděním kolem 272ms, stejně jako u nezatíženého bezdrátového provozu a opět se až do přijímání/odesílání zpráv o velikosti 100kB chová velmi stabilně, dochází k nárůstu cca 50ms. Od této zlomové hranice 100kB, zpoždění znova prudce stoupá stejně jako u předchozího testu, až na hodnotu téměř 1sec, což je pouze o 100ms větší zpoždění, než u nezatížené bezdrátové sítě. Nutno podotknout, že se tento protokol i za takovéto situace chová velmi spolehlivě a rychle, např. MQTT protokol v zatížené bezdrátové síti se potýkal se ztrátou zpráv už při velikosti 100kB.

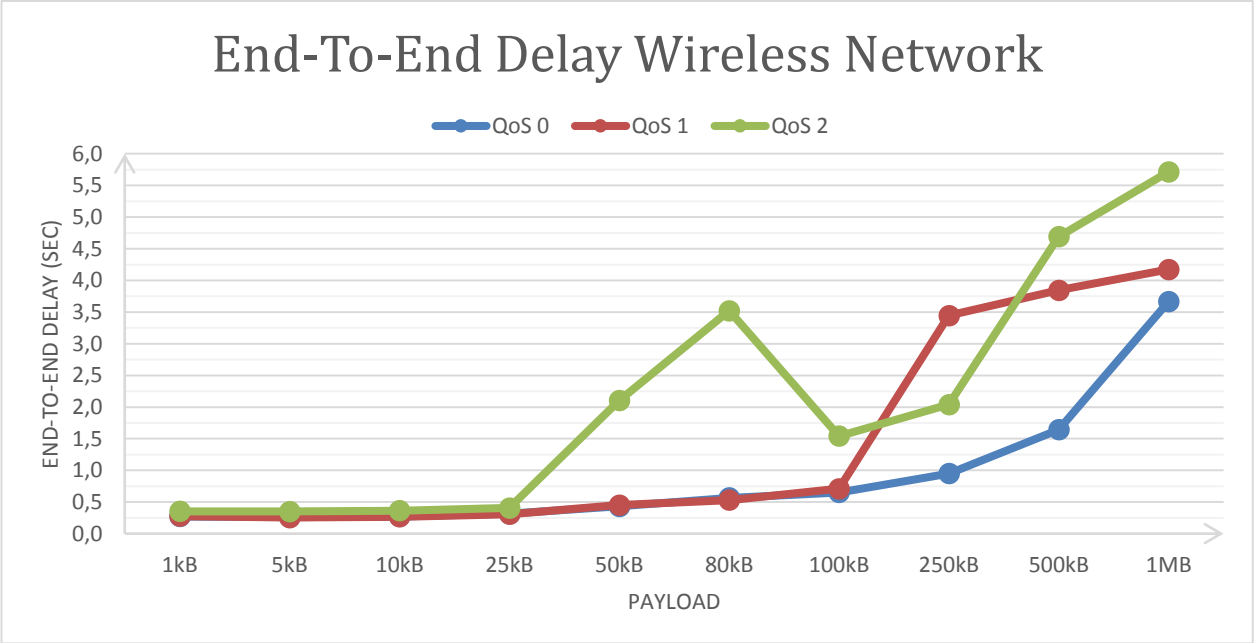


4.3 AMQP

Graf uvedený níže, znázorňuje analýzu zachyceného přenosu dat přes Fast Ethernet pro všechny tři úrovně QoS protokolu AMQP. Jelikož AMQP protokol obsahuje stejně jako MQTT protokol tři typy QoS, budu jej s ním nejvíce srovnávat, ale nejprve musím upozornit na skutečnost, že časová osa grafu je v milisekundách, což značí, že AMQP protokol je rozhodně rychlejší než MQTT, neboť jeho časová osa je uvedena jednotkách sekund. Nejprve se však vrátím k samotnému AMQP. Z grafu je patrné, že protokol se v drátové síti chová stabilně, všechny tři typy QoS si udržují relativně stejné zpoždění až po zprávy o velikosti 250kB, toto zpoždění se pohybuje přibližně od 170ms do 320ms. Od této zlomové hranice 250kB, zpoždění narůstá vzhůru pro všechny tři QoS, přičemž se vyšplhá na hodnoty 430ms až 530ms pro odeslané zprávy o velikosti 1MB. QoS druhé úrovně dle předpokladu splňuje skutečnost, že díky 4-way handshake principu je skutečně nejpomalejší a má největší zpoždění. Což ovšem předpoklady vyvrací je QoS nulté úrovně, od kterého se očekávalo, že bude nejrychlejší, nicméně se zdá, že v tomto měření je jakýmsi kompromisem mezi QoS úrovně 1 a 2. Dalo by se říci, že zpoždění QoS 0 a 1 má téměř identický začátek i konec, a sice oba začínají na 185ms a končí na 435ms, nicméně průběh je odlišný. Tuto skutečnost mohly ovlivnit ztráty na kabelovém vedení, či momentální zatížení sítě. MQTT protokol má ve srovnání s AMQP protokolem mnohem horší výsledky, v podstatě by se dalo pouze říci o QoS 0 a 1, že začínají na přibližně stejné velikosti zpoždění. V konečném důsledku je MQTT protokol téměř 25krát pomalejší, nicméně u obou je zlomovou hranicí velikosti přenášených zpráv 250kB. AMQP by mohlo v drátové síti také konkurovat ZeroMQ protokolu, nicméně v konečném důsledku je nepatrně pomalejší, a sice přibližně o 100ms, zlomová hranice 250kB zůstává. Na závěr bych k tomuto protokolu dodala, že v případě přenosu pro Fast Ethernet, je pro běžné účely nepodstatné kterou úroveň QoS pro přenos použijeme, neboť rozdíly ve výsledcích zpoždění jsou zanedbatelné.

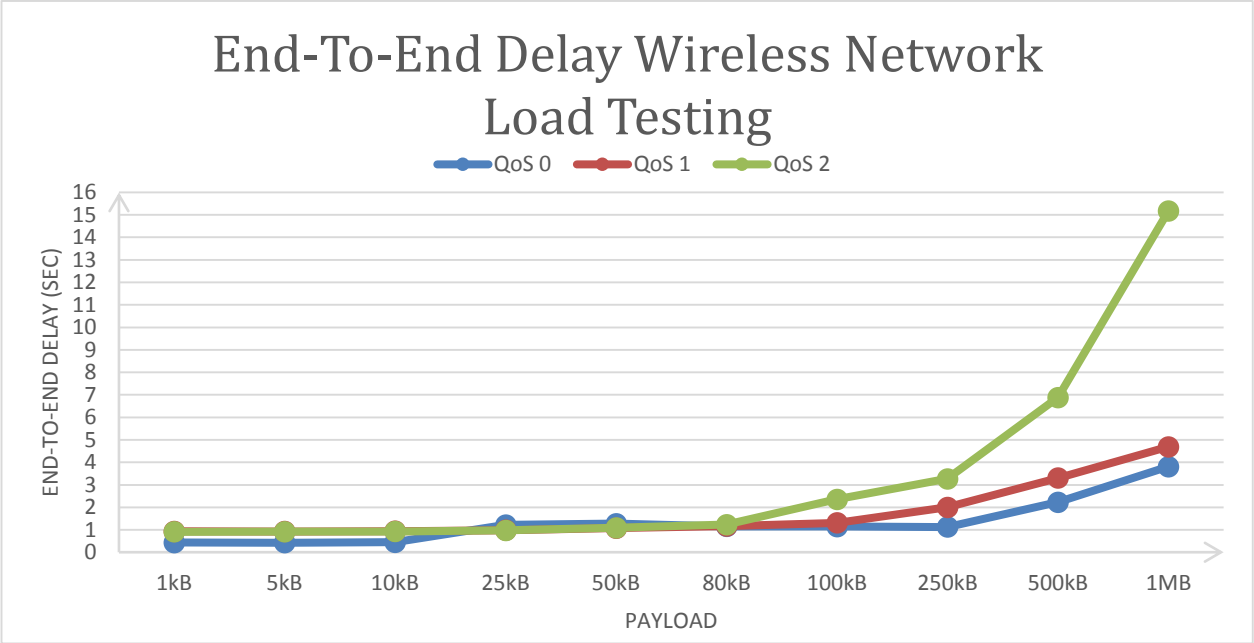


Graf uveden na další straně, znázorňuje analýzu zachyceného provozu pro bezdrátový přenos pro všechny tři úrovně QoS protokolu AMQP. Zdůrazňuji, že vzhledem k nárůstu zpoždění je graf uveden v jednotkách sekund. Rozbor grafu je popsán na další straně.



I po několika uskutečněných měřeních, se AMQP protokol v bezdrátové síti zachoval v náhodné situaci nepředvídatelně. S jistotou můžeme říci, že ve srovnáním s předchozím měřením došlo k podstatnému nárůstu zpoždění. Ačkoliv, jsou časové hodnoty zpoždění pro všechny tři QoS až do velikosti odeslaných zpráv o 25kB téměř identické a pohybují se kolem 250ms, zpoždění QoS-2 v tomto zlomovém bodu začíná velmi prudce stoupat až na zpoždění o velikosti 3,5sec ve vrcholu 80kB. Vzhledem k tomu, že v tomto stoupání plynule nepokračuje, přikládám tento jev za vinu momentálnímu chvilkovému zatížení sítě. Dále zpoždění QoS-2 se zvětšující se velikostí obsahu zpráv prudce stoupá až na hodnotu přibližně 5,7sec, což je ve srovnání s Fast Ethernet přenosem o 5sec více. Nicméně tento graf skutečně splňuje předpoklad spolehlivého, ač pomalejšího QoS úrovně 2 a rychlého, bohužel už ne zrovna spolehlivého QoS úrovně 0. Z grafu lze vidět, že v bezdrátové síti si QoS 0 a 1 silně konkurují a jejich hodnoty jsou téměř identické až po zprávy o velikosti 100kB, v tomto zlomovém bodě zpoždění QoS-1 prudce stoupá až na hodnotu více než 4sec. Jako nejrychlejší se v tomto měření ukázalo být QoS úrovně 0, jehož vývoj zpoždění je zde nejplynulejší a končí při velikosti zpráv 1MB na cca 3,6sec. Ke křivce znázorňující průběh zpoždění QoS-2 bych poznamenala, že toto QoS se jeví v protokolu AMQP, jako velmi náchylné k vnějším vlivům okolí jako je např. momentální krátkodobé zatížení sítě, které se okamžité na zpoždění tohoto QoS projeví. Nicméně v tomto testu v bezdrátové síti už AMQP protokol protokolu ZeroMQ konkurovat nemůže a je až 5krát pomalejší než ZeroMQ, který dosahuje i v bezdrátovém přenosu zpoždění pod 1sec. Avšak AMQP předčil, co se rychlosti týče, opět protokol MQTT a to až o šestinásobek.

Poslední níže uvedený graf pro MQTT protokol, se vztahuje k testu C, který jak již bylo zmíněno, spočívá v zachycení vlivu zatížení bezdrátové sítě na přenos zpráv. Zdůrazňuji, že vzhledem k nárůstu zpoždění je graf opět uveden v jednotkách sekund. Rozbor grafu je popsán na další straně.



Výsledky zátěžového testu pro protokol AMQP dopadly překvapivě. Tak jako u jiných protokolů došlo samozřejmě k nárůstu časových hodnot zpoždění, nicméně tento nárůst je ve srovnání s hodnotami předchozího testu pouze trojnásobný pro všechny tři úrovně QoS. Nicméně v konečném důsledku se AMQP pro QoS 0 a 1 ukázal být jako stabilní a zátěž se na závěrečném zpoždění nijak razantně neprojevila, rozdíl činil 0,5sec. Nejvíce se zatížení sítě projevilo na QoS úrovně druhé, které i v konečném důsledku činilo nárůst zpoždění 3krát větší než v předchozím testu, a sice zpoždění dosahovalo až hodnoty přibližně 15sec. Tento test splnil předpoklady a příjemně překvapil, neboť AMQP protokol si dokázal poradit se zátěžovým testem podstatně lépe než MQTT protokol, nicméně ZeroMQ opět nepřekonal, neboť ten se se svým časovým zpožděním opět vtěsnal do méně než 1sec. Rozhodně můžeme potvrdit, že nedošlo k žádným ztrátám paketů/zpráv, jakož tomu bylo v případě MQTT protokolu. AMQP zátěžový test až do zlomové hranice 80-100kB snáší velmi stabilně, bez větších výkyvů, ty nastávají s příchodem zpráv o velikosti 500kB a výše. Opět lze vidět, že QoS 0 a 1 si vedou v testu velmi vyrovnaně a rozdíly ve zpoždění jsou nepatrné, v konečném důsledku činí necelou 1sec.

Závěr

V teoretické části této práce byly popsány komunikační protokoly současnosti i budoucnosti, nejzajímavějším a nejperspektivnějším z nich byla věnována praktická část práce. Práce splnila mou osobní počáteční vizi, s kterou jsem k ní přistupovala a překvapila mě, co se dosažených výsledků týče. V praktické části práce jsem se zaobírala konfigurací, implementací a testováním tří protokolů: MQTT, ZeroMQ a AMQP. Má původní představa obsahovala i implementaci a testování protokolu XMPP, což je jeden z nejznámějších protokolů pro Instant Messaging. Nicméně, implementaci tohoto protokolu hodnotím, jako velmi náročnou, co se času i znalostí týče, a i to je jeden z hlavních hodnotících faktorů a důvodů, proč tento protokol bych dále nedoporučila. Naopak, bych vřele doporučila, jak ke studijním tak i k experimentálním účelům protokol ZeroMQ, jehož implementace byla nejméně časově náročná a přívětivě se s tímto protokolem pracovalo. Také si dovoluji říct, že ve všech testech dosáhl nejlepších výsledků, jedná se o velmi stabilní protokol i v zatížené bezdrátové síti a domnívám se, že tento protokol najde ještě mnoho uplatnění v oblasti internetu věcí. Nemile mne překvapil protokol MQTT, který se v zatížené bezdrátové síti projevil jako velmi nespolehlivý a ztrátový, a obecně ve všech testech této práce dopadl nejhůře, což jsem od tak známého a často zmiňovaného protokolu nepředpokládala.

Tato práce z hlediska dalšího vývoje má ještě mnoho možností. Jednak se stále objevují a vytvářejí nové komunikační protokoly, nebo naopak jsou vyvíjeny nové a lepší verze protokolů současných, tudíž práci by bylo možné rozšířit o další implementace. Dále by určitě stálo za zmínku podrobnější a rozsáhlejší testování, především testování implementací protokolů na různých přenosových rychlostech a jiných přenosových technologiích. Pokud by testování probíhalo na vyšších přenosových rychlostech, bylo by možné přenášet i obsahově mnohem větší zprávy. Práci by bylo možné rozšířit o implementace protokolů ve vícero programovacích jazycích a testovat tak vhodné kombinace programovacího jazyka a zprostředkovatele broker serveru. Analýza komunikačních protokolů by mohla také spočívat v testování ztrátovosti komunikačních protokolů, neboť pro měření spolehlivosti protokolu je důležitým faktorem poměr zahozených paketů k celkovému počtu paketů vyslaných.

Závěrem chci podotknout, že tato práce vychází především z teoreticky nastudovaných předpokladů a simulací, nicméně většina dosažených výsledků splnila předpoklady a několikanásobné měření a testování tyto výsledky potvrzuje. Jednotlivá měření byly prováděny opakovaně, výsledky každého měření jsou podloženy téměř dvěma tisíci hodnot. Budu ráda, pokud tato práce bude někomu inspirací, nebo jen poslouží jako zdroj informací a podklad pro další studium.

Použitá literatura

[1] BURIAN, Pavel. Internet inteligentních aktivit. 1. vyd. Praha: Grada Publishing, a.s., 2014. 336 s. ISBN 978-80-247-5137-5. Kapitola 9, Internet věcí, inteligentních výrobků a technologií

[2] ASHTON, Kevin. RFID Journal[online]. 2009 [cit. 2015-11-23]. Dostupné z: http://www.rfidjournal.com/articles/view?4986

[3] SUNDMEAKER, Harald. Vision and Challenges for Realising the Internet of Things. Luxembourg: Publications Office of the European Union, 2010. ISBN 978-92-79-15088-3. 236 s.

[4] IoT v roce 2020: 25 miliard věcí připojených k internetu. BusinessIT.cz[online]. 2015 [cit. 2015-11-23]. Dostupné z: http://www.businessit.cz/cz/iot-v-roce-2020-25-miliard-veci-pripojenych-k-internetu.php

[5] Tapping into the Internet of Things. IBM developerWorks[online]. 2015 [cit. 2015-11-24]. Dostupné z: http://www.ibm.com/developerworks/library/iot-key-concepts/index.html

[6] TOMÁNEK, Filip. Internet věcí. Přehnaný hype nebo svěží vítr pro business?. Trask.cz[online]. 2015 [cit. 2015.11-24]. Dostupné z: http://www.trask.cz/publikace/zn-89-internet-veci-prehnany-hype-nebo-svezi-vitr-pro-business/

[7] WDT s.r.o.. Co je RFID?. RFID-EPC.cz[online]. 2014 [cit. 2015-11-24]. Dostupné z: http://www.rfid-epc.cz/co-je-rfid/historie-rfid/

[8] SOMMEROVÁ, Martina. Základy RFID technologií. vsb.cz[online]. 2015 [cit. 2015-11-24]. Dostupné z: http://rfid.vsb.cz/export/sites/rfid/cs/informace/RFID_pro_Logistickou_akademii.pdf

[9] VOJTĚCH, L. RFID - technologie pro internet věcí. fel.cvut.cz[online]. 2009 [cit. 2015-11-24]. Dostupné z: http://access.fel.cvut.cz/view.php?cisloclanku=2009020001

[10] Základní schéma komunikace v RFID. In: access.fel.cvut.cz[online]. 2009 [cit. 2015-11-24]. Dostupné z: http://access.fel.cvut.cz/storage/200902121056_obr_1.png

[11] Technologie NFC: Placení mobilním telefonem. Chip.cz[online]. 2012 [cit. 2015-12-15]. Dostupné z: http://www.chip.cz/casopis-chip/earchiv/vydani/r-2012/chip-05-2012/technologie-nfc/

[12] Secure element: klíč k mobilním platbám. Nearfield.cz[online]. 2012 [cit. 2015-12-15]. Dostupné z: http://nearfield.cz/clanky/secure-element-klic-k-mobilnim-platbam-20

[13] Co je to NFC a co umí?. Nearfield.cz[online]. 2012 [cit. 2015-12-15]. Dostupné z: http://nearfield.cz/co-je-nfc

[14] PARRISH, Kevin. ZigBee, Z-Wave, Thread and WeMo: What's the difference?. Tomsguide.com[online]. 2015[cit. 2015-12-15]. Dostupné z: http://www.tomsguide.com/us/smart-home-wireless-network-primer,news-21085.html

[15] CHAKRABARTI, Annirudha. InfoQ.com[online]. 2015 [cit. 2015-12-15]. Dostupné z: http://www.infoq.com/articles/thread-protocol-for-home-automation

[16] Thread Technology. Threadgroup.org[online]. 2015 [cit. 2015-12-15]. Dostupné z: http://threadgroup.org/Technology.aspx

[17] What is ZigBee?. ZigBeeAlliance[online]. 2015[cit. 2015-12-15]. Dostupné z: http://www.zigbee.org/what-is-zigbee/

[18] PAVLIS, Jakub. ZigBee - když je pomalejší síť výhodnější. Notebook.cz[online]. 2011 [cit. 2015-12-15]. Dostupné z: http://notebook.cz/clanky/technologie/2011/ZigBee

[19] Funkce Z-Wave. IntelioBOX[online]. 2015[cit. 2015-12-15]. Dostupné z: http://www.inteliobox.com/cz/z-wave/funkce-z-wave

[20] Tabulka. In: Tomsguide.com[online]. 2015 [cit. 2015-12-15]. Dostupné z: http://blog.orvibo.com/enblog/wp-content/uploads/2015/11/tab.jpg

[21] WALSH, Terry. What's the Difference Between Google Weave and Nest Weave?. WeGotServed[online]. 2015[cit. 2015-12-17]. Dostupné z: http://www.wegotserved.com/2015/10/03/whats-the-difference-between-google-weave-and-nest-weave/

[22] How Weave works. Weave[online]. 2015[cit. 2015-12-17]. Dostupné z: http://docs.weave.works/weave/latest_release/how-it-works.html

[23] Shinho Lee; Hyeonwoo Kim; Dong-Kweon Hong; Hongtaek Ju, "Correlation analysis of MQTT loss and delay according to QoS level," in Information Networking (ICOIN), 2013 International Conference on , vol., no., pp.714-717, 28-30 Jan. 2013[cit. 2015-12-17]. doi: 10.1109/ICOIN.2013.6496715. Dostupné z: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6496715&isnumber=6496336

[24] IBM, Eurotech. MQTT V3.1 Protocol Specification. IBM.com[online]. 2010[cit. 2015-12-17]. Dostupné z: http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html

[25] OASIS: protokolem internetu věcí bude MQTT. ComputerWorld.cz[online]. 2013[cit. 2015-12-17]. Dostupné z: http://computerworld.cz/technologie/oasis-protokolem-internetu-veci-bude-mqtt-49824

[26] Benchmark of MQTT servers. MQTT.org[online formát PDF]. 2015[cit. 2015-12-17]. Dostupné z: http://www.scalagent.com/IMG/pdf/Benchmark_MQTT_servers-v1-1.pdf

[27] MESNIL, Jeff. Mobile and Web Messaging. 1. vyd. USA: O'Reilly Media, Inc., 2014. 183s. ISBN 978-1-491-94480-6

[28] STOMP Protocol Specification, V.1.2. Stomp.github.io[online]. 2015[cit. 2015-12-17]. Dostupné z: https://stomp.github.io/stomp-specification-1.2.html

[29] Figure2. In: IBM developerWorks[online]. 2009[cit. 2015-12-26]. Dostupné z: http://www.ibm.com/developerworks/library/x-xmppintro/figure2.gif

[30] JONES, M. Tim. Meet the XMPP. IBM developerWorks[online]. 2009[cit. 2015-12-26]. Dostupné z: http://www.ibm.com/developerworks/library/x-xmppintro/

[31] Jabber. Jabber.cz[online]. 2015[cit. 2015-12-26]. Dostupné z: http://www.jabber.cz/wiki/Jabber

[32] Extensible Messaging and Presence Protocol. Wikipedie.cz[online]. 2015[cit. 2015-12-26]. Dostupné z: https://cs.wikipedia.org/wiki/Extensible_Messaging_and_Presence_Protocol

[33] AMQP is the Internet protocol dor Business Messaging. AMQP.org[online]. 2015[cit. 2015-12-30]. Dostupné z: http://www.amqp.org/about/what

[34] Features. AMQP.org[online]. 2015[cit. 2015-12-30]. Dostupné z: http://www.amqp.org/product/features

[35] Advanced Mesagge Queuing Protocol. Wikipedia.org[online]. 2015[cit. 2015-12-30]. Dostupné z: https://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol

[36] Architecture. AMPQ.org[online]. 2015[cit. 2015-12-30]. Dostupné z: http://www.amqp.org/product/architecture

[37] ZeroMQ. Wikipedia.org[online]. 2015[cit. 2015-12-30]. Dostupné z: https://en.wikipedia.org/wiki/%C3%98MQ

[38] AKGUL, Faruk. ZeroMQ. 1.vyd. UK: Packt Publishing, 2013. 108s. ISBN 978-1-78216-104-2

[39] HINTJENS, Pieter. ØMQ - The Guide. ZeroMQ.org[online]. 2014[cit. 2016-04-18]. Dostupné z: http://zguide.zeromq.org/page:all

[40] Pivotal Software. RabbitMQ API Guide. RabbitMQ.com[online]. 2016[cit. 2016-04-25]. Dostupné z: http://www.rabbitmq.com/api-guide.html

Seznam příloh

Přílohy na CD/DVD.

Adresářová struktura přiloženého CD/DVD:

- AMQP - AMQP_namerene_hodnoty.xlsx
  - AMQP_source_code.zip
  - AMQPConsumer-1.0-jar-with-dependencies.jar
  - AMQPProducer-1.0-jar-with-dependencies.jar

- MQTT - MQTT_namerene_hodnoty.xlsx
  - MQTT_source_code.zip

- ZeroMQ - ZeroMQ_namerene_hodnoty.xlsx
  - ZeroMQ_source_code.zip
  - ZeroMQ-1.0-jar-with-dependencies.jar
  - ZeroMQServer-1.0-jar-with-dependencies.jar